

# AN INFORMATION-THEORETIC FRAMEWORK FOR FAST AND ROBUST UNSUPERVISED LEARNING VIA NEURAL POPULATION INFOMAX

**Wentao Huang & Kechen Zhang**

Department of Biomedical Engineering  
Johns Hopkins University School of Medicine  
Baltimore, MD 21205, USA  
{whuang21, kzhang4}@jhmi.edu

## ABSTRACT

A framework is presented for unsupervised learning of representations based on infomax principle for large-scale neural populations. We use an asymptotic approximation to the Shannon’s mutual information for a large neural population to demonstrate that a good initial approximation to the global information-theoretic optimum can be obtained by a hierarchical infomax method. Starting from the initial solution, an efficient algorithm based on gradient descent of the final objective function is proposed to learn representations from the input datasets, and the method works for complete, overcomplete, and undercomplete bases. As confirmed by numerical experiments, our method is robust and highly efficient for extracting salient features from input datasets. Compared with the main existing methods, our algorithm has a distinct advantage in both the training speed and the robustness of unsupervised representation learning. Moreover, the method is readily extended to train deep structure networks by unsupervised or supervised learning.

## 1 INTRODUCTION

How to discover the unknown structures in data is a key task for machine learning. Learning good representations from observed data is important because a clearer description may help reveal the underlying structures. Representation learning has drawn considerable attention in recent years (Bengio et al., 2013). One category of algorithms for unsupervised learning of representations is based on probabilistic models (Lewicki & Sejnowski, 2000; Hinton & Salakhutdinov, 2006; Lee et al., 2008), such as maximum likelihood (ML) estimation, maximum a posteriori (MAP) probability estimation, and related methods. Another category of algorithms is based on reconstruction error or generative criterion (Olshausen & Field, 1996; Aharon et al., 2006; Vincent et al., 2010; Mairal et al., 2010; Goodfellow et al., 2014), and the objective functions usually involve squared errors with additional constraints. Sometimes the reconstruction error or generative criterion may also have a probabilistic interpretation (Olshausen & Field, 1997; Vincent et al., 2010).

Shannon’s information theory is a powerful tool for description of stochastic systems and could be utilized to provide a characterization for good representations (Vincent et al., 2010). However, computational difficulties associated with Shannon’s mutual information (MI) (Shannon, 1948) have hindered its wider applications. The Monte Carlo (MC) sampling (Yarrow et al., 2012) is a convergent method for estimating MI with arbitrary accuracy, but its computational inefficiency makes it unsuitable for difficult optimization problems especially in the cases of high-dimensional input stimuli and large population networks. Bell and Sejnowski (Bell & Sejnowski, 1995; 1997) have directly applied the infomax approach (Linsker, 1988) to independent component analysis (ICA) of data with independent non-Gaussian components assuming additive noise, but their method requires that the number of outputs be equal to the number of inputs. The extensions of ICA to overcomplete or undercomplete bases incur increased algorithm complexity and difficulty in learning of parameters (Lewicki & Sejnowski, 2000; Kreutz-Delgado et al., 2003; Karklin & Simoncelli, 2011).

Since Shannon MI is closely related to ML and MAP (Huang & Zhang, 2016), the algorithms of representation learning based on probabilistic models should be amenable to information-theoretic treatment. Representation learning based on reconstruction error could be accommodated also by information theory, because the inverse of Fisher information (FI) is the Cramér-Rao lower bound on the mean square decoding error of any unbiased decoder (Rao, 1945). Hence minimizing the reconstruction error potentially maximizes a lower bound on the MI (Vincent et al., 2010).

Related problems arise also in neuroscience. It has long been suggested that the real nervous systems might approach an information-theoretic optimum for neural coding and computation (Barlow, 1961; Atick, 1992; Borst & Theunissen, 1999). However, in the cerebral cortex, the number of neurons is huge, with about  $10^5$  neurons under a square millimeter of cortical surface (Carlo & Stevens, 2013). It has often been computationally intractable to precisely characterize information coding and processing in large neural populations.

To address all these issues, we present a framework for unsupervised learning of representations in a large-scale nonlinear feedforward model based on infomax principle with realistic biological constraints such as neuron models with Poisson spikes. First we adopt an objective function based on an asymptotic formula in the large population limit for the MI between the stimuli and the neural population responses (Huang & Zhang, 2016). Since the objective function is usually nonconvex, choosing a good initial value is very important for its optimization. Starting from an initial value, we use a hierarchical infomax approach to quickly find a tentative global optimal solution for each layer by analytic methods. Finally, a fast convergence learning rule is used for optimizing the final objective function based on the tentative optimal solution. Our algorithm is robust and can learn complete, overcomplete or undercomplete basis vectors quickly from different datasets. Experimental results showed that the convergence rate of our method was significantly faster than other existing methods, often by an order of magnitude. More importantly, the number of output units processed by our method can be very large, much larger than the number of inputs. As far as we know, no existing model can easily deal with this situation.

## 2 METHODS

### 2.1 APPROXIMATION OF MUTUAL INFORMATION FOR NEURAL POPULATIONS

Suppose the input  $\mathbf{x}$  is a  $K$ -dimensional vector,  $\mathbf{x} = (x_1, \dots, x_K)^T$ , the outputs of  $N$  neurons are denoted by a vector,  $\mathbf{r} = (r_1, \dots, r_N)^T$ , where we assume  $N$  is large, generally  $N \gg K$ . We denote random variables by upper case letters, e.g., random variables  $X$  and  $R$ , in contrast to their vector values  $\mathbf{x}$  and  $\mathbf{r}$ . The MI between  $X$  and  $R$  is defined by  $I(X; R) = \left\langle \ln \frac{p(\mathbf{x}|\mathbf{r})}{p(\mathbf{x})} \right\rangle_{\mathbf{r}, \mathbf{x}}$ , where  $\langle \cdot \rangle_{\mathbf{r}, \mathbf{x}}$  denotes the expectation with respect to the probability density function (PDF)  $p(\mathbf{r}, \mathbf{x})$ .

Our goal is to maximize MI  $I(X; R)$  by finding the optimal PDF  $p(\mathbf{r}|\mathbf{x})$  under some constraint conditions, assuming that  $p(\mathbf{r}|\mathbf{x})$  is characterized by a noise model and activation functions  $f(\mathbf{x}; \boldsymbol{\theta}_n)$  with parameters  $\boldsymbol{\theta}_n$  for the  $n$ -th neuron ( $n = 1, \dots, N$ ). In other words, we optimize  $p(\mathbf{r}|\mathbf{x})$  by solving for the optimal parameters  $\boldsymbol{\theta}_n$ . Unfortunately, it is intractable in most cases to solve for the optimal parameters that maximizes  $I(X; R)$ . However, if  $p(\mathbf{x})$  and  $p(\mathbf{r}|\mathbf{x})$  are twice continuously differentiable for almost every  $\mathbf{x} \in \mathbb{R}^K$ , then for large  $N$  we can use an asymptotic formula to approximate the true value of  $I(X; R)$  with high accuracy (Huang & Zhang, 2016):

$$I(X; R) \simeq I_G = \frac{1}{2} \left\langle \ln \left( \det \left( \frac{\mathbf{G}(\mathbf{x})}{2\pi e} \right) \right) \right\rangle_{\mathbf{x}} + H(X), \quad (1)$$

where  $\det(\cdot)$  denotes the matrix determinant and  $H(X) = -\langle \ln p(\mathbf{x}) \rangle_{\mathbf{x}}$  is the stimulus entropy,

$$\mathbf{G}(\mathbf{x}) = \mathbf{J}(\mathbf{x}) + \mathbf{P}(\mathbf{x}), \quad (2)$$

$$\mathbf{J}(\mathbf{x}) = - \left\langle \frac{\partial^2 \ln p(\mathbf{r}|\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T} \right\rangle_{\mathbf{r}|\mathbf{x}}, \quad (3)$$

$$\mathbf{P}(\mathbf{x}) = - \frac{\partial^2 \ln p(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T}. \quad (4)$$

Assuming independent noises in neuronal responses, we have  $p(\mathbf{r}|\mathbf{x}) = \prod_{n=1}^N p(r_n|\mathbf{x}; \boldsymbol{\theta}_n)$ , and the Fisher information matrix becomes  $\mathbf{J}(\mathbf{x}) \approx N \sum_{k=1}^{K_1} \alpha_k \mathbf{S}(\mathbf{x}; \boldsymbol{\theta}_k)$ , where  $\mathbf{S}(\mathbf{x}; \boldsymbol{\theta}_k) =$

$\left\langle \frac{\partial \ln p(r|\mathbf{x}; \boldsymbol{\theta}_k)}{\partial \mathbf{x}} \frac{\partial \ln p(r|\mathbf{x}; \boldsymbol{\theta}_k)}{\partial \mathbf{x}^T} \right\rangle_{r|\mathbf{x}}$  and  $\alpha_k > 0$  ( $k = 1, \dots, K_1$ ) is the population density of parameter  $\boldsymbol{\theta}_k$ , with  $\sum_{k=1}^{K_1} \alpha_k = 1$ , and  $1 \leq K_1 \leq N$  (see Appendix A.1 for details). Since the cerebral cortex usually forms functional column structures and each column is composed of neurons with the same properties (Hubel & Wiesel, 1962), the positive integer  $K_1$  can be regarded as the number of distinct classes in the neural population.

Therefore, given the activation function  $f(\mathbf{x}; \boldsymbol{\theta}_k)$ , our goal becomes to find the optimal population distribution density  $\alpha_k$  of parameter vector  $\boldsymbol{\theta}_k$  so that the MI between the stimulus  $\mathbf{x}$  and the response  $\mathbf{r}$  is maximized. By Eq. (1), our optimization problem can be stated as follows:

$$\text{minimize } Q_G[\{\alpha_k\}] = -\frac{1}{2} \langle \ln(\det(\mathbf{G}(\mathbf{x}))) \rangle_{\mathbf{x}}, \quad (5)$$

$$\text{subject to } \sum_{k=1}^{K_1} \alpha_k = 1, \alpha_k > 0, \forall k = 1, \dots, K_1. \quad (6)$$

Since  $Q_G[\{\alpha_k\}]$  is a convex function of  $\{\alpha_k\}$  (Huang & Zhang, 2016), we can readily find the optimal solution for small  $K$  by efficient numerical methods. For large  $K$ , however, finding an optimal solution by numerical methods becomes intractable. In the following we will propose an alternative approach to this problem. Instead of directly solving for the density distribution  $\{\alpha_k\}$ , we optimize the parameters  $\{\alpha_k\}$  and  $\{\boldsymbol{\theta}_k\}$  simultaneously under a hierarchical infomax framework.

## 2.2 HIERARCHICAL INFOMAX

For clarity, we consider neuron model with Poisson spikes although our method is easily applicable to other noise models. The activation function  $f(\mathbf{x}; \boldsymbol{\theta}_n)$  is generally a nonlinear function, such as sigmoid and rectified linear unit (ReLU) (Nair & Hinton, 2010). We assume that the nonlinear function for the  $n$ -th neuron has the following form:  $f(\mathbf{x}; \boldsymbol{\theta}_n) = \tilde{f}(y_n; \tilde{\boldsymbol{\theta}}_n)$ , where

$$y_n = \mathbf{w}_n^T \mathbf{x}. \quad (7)$$

with  $\mathbf{w}_n$  being a  $K$ -dimensional weights vector,  $\tilde{f}(y_n; \tilde{\boldsymbol{\theta}}_n)$  is a nonlinear function,  $\boldsymbol{\theta}_n = (\mathbf{w}_n^T, \tilde{\boldsymbol{\theta}}_n^T)^T$  and  $\tilde{\boldsymbol{\theta}}_n$  are the parameter vectors ( $n = 1, \dots, N$ ).

In general, it is very difficult to find the optimal parameters,  $\boldsymbol{\theta}_n$ ,  $n = 1, \dots, N$ , for the following reasons. First, the number of output neurons  $N$  is very large, usually  $N \gg K$ . Second, the activation function  $f(\mathbf{x}; \boldsymbol{\theta}_n)$  is a nonlinear function, which usually leads to a nonconvex optimization problem. For nonconvex optimization problems, the selection of initial values often has a great influence on the final optimization results. Our approach meets these challenges by making better use of the large number of neurons and by finding good initial values by a hierarchical infomax method.

We divide the nonlinear transformation into two stages, mapping first from  $\mathbf{x}$  to  $y_n$  ( $n = 1, \dots, N$ ), and then from  $y_n$  to  $\tilde{f}(y_n; \tilde{\boldsymbol{\theta}}_n)$ , where  $y_n$  can be regarded as the membrane potential of the  $n$ -th neuron, and  $\tilde{f}(y_n; \tilde{\boldsymbol{\theta}}_n)$  as its firing rate. As with the real neurons, we assume that the membrane potential is corrupted by noise:

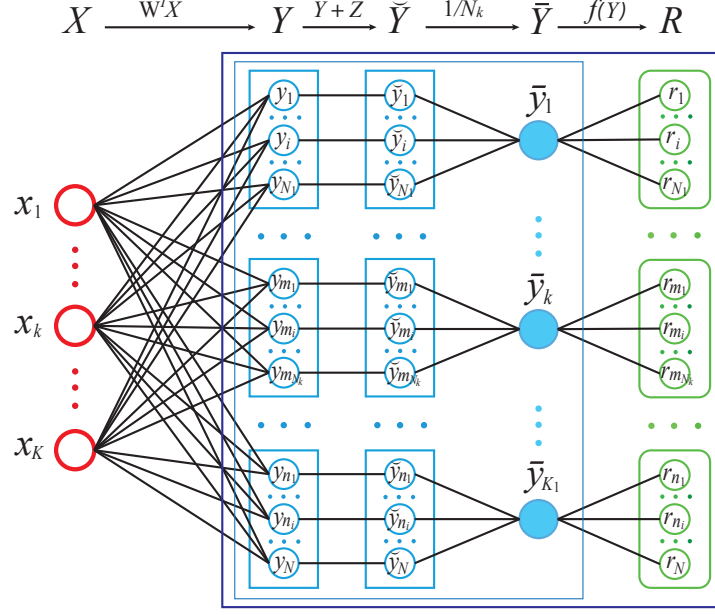
$$\check{Y}_n = Y_n + Z_n, \quad (8)$$

where  $Z_n \sim \mathcal{N}(0, \sigma^2)$  is a normal distribution with mean 0 and variance  $\sigma^2$ . Then the mean membrane potential of the  $k$ -th class subpopulation with  $N_k = N\alpha_k$  neurons is given by

$$\bar{Y}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \check{Y}_{k_n} = Y_k + \bar{Z}_k, k = 1, \dots, K_1, \quad (9)$$

$$\bar{Z}_k \sim \mathcal{N}(0, N_k^{-1} \sigma^2). \quad (10)$$

Define vectors  $\check{\mathbf{y}} = (\check{y}_1, \dots, \check{y}_N)^T$ ,  $\bar{\mathbf{y}} = (\bar{y}_1, \dots, \bar{y}_{K_1})^T$  and  $\mathbf{y} = (y_1, \dots, y_{K_1})^T$ , where  $y_k = \mathbf{w}_k^T \mathbf{x}$  ( $k = 1, \dots, K_1$ ). Notice that  $\check{y}_n$  ( $n = 1, \dots, N$ ) is also divided into  $K_1$  classes, the same as for  $r_n$ . If we assume  $f(\mathbf{x}; \boldsymbol{\theta}_k) = \tilde{f}(\bar{y}_k; \tilde{\boldsymbol{\theta}}_k)$ , i.e. assuming an additive Gaussian noise for  $y_n$  (see Eq. 9), then the random variables  $X$ ,  $Y$ ,  $\check{Y}$ ,  $\bar{Y}$  and  $R$  form a Markov chain, denoted by  $X \rightarrow Y \rightarrow \check{Y} \rightarrow \bar{Y} \rightarrow R$  (see Figure 1), and we have the following proposition (see Appendix A.2).

Figure 1: A neural network interpretation for random variables  $X, Y, \check{Y}, \bar{Y}, R$ .

**Proposition 1.** With the random variables  $X, Y, \check{Y}, \bar{Y}, R$  and Markov chain  $X \rightarrow Y \rightarrow \check{Y} \rightarrow \bar{Y} \rightarrow R$ , the following equations hold,

$$I(X; R) = I(Y; R) \leq I(\check{Y}; R) \leq I(\bar{Y}; R), \quad (11)$$

$$I(X; R) \leq I(X; \bar{Y}) = I(X; \check{Y}) \leq I(X; Y), \quad (12)$$

and for large  $N_k$  ( $k = 1, \dots, K_1$ ),

$$I(\check{Y}; R) \simeq I(\bar{Y}; R) \simeq I(Y; R) = I(X; R), \quad (13)$$

$$I(X; Y) \simeq I(X; \bar{Y}) = I(X; \check{Y}). \quad (14)$$

A major advantage of incorporating membrane noise is that it facilitates finding the optimal solution by using the infomax principle. Moreover, the optimal solution obtained this way is more robust; that is, it discourages overfitting and has a strong ability to resist distortion. With vanishing noise  $\sigma^2 \rightarrow 0$ , we have  $\bar{Y}_k \rightarrow Y_k$ ,  $\tilde{f}(\bar{y}_k; \bar{\theta}_k) \simeq \tilde{f}(y_k; \bar{\theta}_k) = f(\mathbf{x}; \theta_k)$ , so that Eqs. (13) and (14) hold as in the case of large  $N_k$ .

To optimize MI  $I(Y; R)$ , the probability distribution of random variable  $Y$ ,  $p(\mathbf{y})$ , needs to be determined, i.e. maximizing  $I(Y; R)$  about  $p(\mathbf{y})$  under some constraints should yield an optimal distribution:  $p^*(\mathbf{y}) = \arg \max_{p(\mathbf{y})} I(Y; R)$ . Let  $\mathcal{C} = \max_{p(\mathbf{y})} I(Y; R)$  be the channel capacity of neural population coding, and we always have  $I(X; R) \leq \mathcal{C}$  (Huang & Zhang, 2016). To find a suitable linear transformation from  $X$  to  $Y$  that is compatible with this distribution  $p^*(\mathbf{y})$ , a reasonable choice is to maximize  $I(X; \check{Y})$  ( $\leq I(X; Y)$ ), where  $\check{Y}$  is a noise-corrupted version of  $Y$ . This implies minimum information loss in the first transformation step. However, there may exist many transformations from  $X$  to  $\check{Y}$  that maximize  $I(X; \check{Y})$  (see Appendix A.3.1). Ideally, if we can find a transformation that maximizes both  $I(X; \check{Y})$  and  $I(Y; R)$  simultaneously, then  $I(X; R)$  reaches its maximum value:  $I(X; R) = \max_{p(\mathbf{y})} I(Y; R) = \mathcal{C}$ .

From the discussion above we see that maximizing  $I(X; R)$  can be divided into two steps, namely, maximizing  $I(X; \check{Y})$  and maximizing  $I(Y; R)$ . The optimal solutions of  $\max I(X; \check{Y})$  and  $\max I(Y; R)$  will provide a good initial approximation that tend to be very close to the optimal solution of  $\max I(X; R)$ .

Similarly, we can extend this method to multilayer neural population networks. For example, a two-layer network with outputs  $R^{(1)}$  and  $R^{(2)}$  form a Markov chain,  $X \rightarrow \tilde{R}^{(1)} \rightarrow R^{(1)} \rightarrow \bar{R}^{(1)} \rightarrow$



$R^{(2)}$ , where random variable  $\tilde{R}^{(1)}$  is similar to  $Y$ , random variable  $R^{(1)}$  is similar to  $\check{Y}$ , and  $\bar{R}^{(1)}$  is similar to  $\bar{Y}$  in the above. Then we can show that the optimal solution of  $\max I(X; R^{(2)})$  can be approximated by the solutions of  $\max I(X; R^{(1)})$  and  $\max I(\tilde{R}^{(1)}; R^{(2)})$ , with  $I(\tilde{R}^{(1)}; R^{(2)}) \simeq I(\bar{R}^{(1)}; R^{(2)})$ .

More generally, consider a highly nonlinear feedforward neural network that maps the input  $\mathbf{x}$  to output  $\mathbf{z}$ , with  $\mathbf{z} = F(\mathbf{x}; \boldsymbol{\theta}) = h_L \circ \dots \circ h_1(\mathbf{x})$ , where  $h_l$  ( $l = 1, \dots, L$ ) is a linear or nonlinear function (Montufar et al., 2014). We aim to find the optimal parameter  $\boldsymbol{\theta}$  by maximizing  $I(X; Z)$ . It is usually difficult to solve the optimization problem when there are many local extrema for  $F(\mathbf{x}; \boldsymbol{\theta})$ . However, if each function  $h_l$  is easy to optimize, then we can use the hierarchical infomax method described above to get a good initial approximation to its global optimization solution, and go from there to find the final optimal solution. This information-theoretic consideration from the neural population coding point of view may help explain why deep structure networks with unsupervised pre-training have a powerful ability for learning representations.

### 2.3 THE OBJECTIVE FUNCTION

The optimization processes for maximizing  $I(X; \check{Y})$  and maximizing  $I(Y; R)$  are discussed in detail in Appendix A.3. First, by maximizing  $I(X; \check{Y})$  (see Appendix A.3.1 for details), we can get the optimal weight parameter  $\mathbf{w}_k$  ( $k = 1, \dots, K_1$ , see Eq. 7) and its population density  $\alpha_k$  (see Eq. 6) which satisfy

$$\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_{K_1}] = a \mathbf{U}_0 \boldsymbol{\Sigma}_0^{-1/2} \mathbf{C}, \quad (15)$$

$$\alpha_1 = \dots = \alpha_{K_1} = K_1^{-1}, \quad (16)$$

where  $a = \sqrt{K_1 K_0^{-1}}$ ,  $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_{K_1}] \in \mathbb{R}^{K_0 \times K_1}$ ,  $\mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}$ ,  $\mathbf{I}_{K_0}$  is a  $K_0 \times K_0$  identity matrix with integer  $K_0 \in [1, K]$ , the diagonal matrix  $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{K_0 \times K_0}$  and matrix  $\mathbf{U}_0 \in \mathbb{R}^{K \times K_0}$  are given in (A.44) and (A.45), with  $K_0$  given by Eq. (A.52). Matrices  $\boldsymbol{\Sigma}_0$  and  $\mathbf{U}_0$  can be obtained by  $\boldsymbol{\Sigma}$  and  $\mathbf{U}$  with  $\mathbf{U}_0^T \mathbf{U}_0 = \mathbf{I}_{K_0}$  and  $\mathbf{U}_0 \boldsymbol{\Sigma}_0 \mathbf{U}_0^T \approx \mathbf{U} \boldsymbol{\Sigma} \mathbf{U}^T \approx \langle \mathbf{x} \mathbf{x}^T \rangle_{\mathbf{x}}$  (see Eq. A.23). The optimal weight parameter  $\mathbf{w}_k$  (15) means that the input variable  $\mathbf{x}$  must first undergo a whitening-like transformation  $\hat{\mathbf{x}} = \boldsymbol{\Sigma}_0^{-1/2} \mathbf{U}_0^T \mathbf{x}$ , and then goes through the transformation  $\mathbf{y} = a \mathbf{C}^T \hat{\mathbf{x}}$ , with matrix  $\mathbf{C}$  to be optimized below. Note that weight matrix  $\mathbf{W}$  is low rank with  $\text{rank}(\mathbf{W}) = \min(K_0, K_1)$ , and its low dimensionality helps reduce overfitting during training (see Appendix A.3.1).

By maximizing  $I(Y; R)$  (see Appendix A.3.2), we further solve for the the optimal parameters  $\tilde{\boldsymbol{\theta}}_k$  for the nonlinear functions  $\tilde{f}(y_k; \tilde{\boldsymbol{\theta}}_k)$ ,  $k = 1, \dots, K_1$ . Finally, the objective function for our optimization problem (Eqs. 5 and 6) turns into (see Appendix A.3.3 for details):

$$\text{minimize } Q[\mathbf{C}] = -\frac{1}{2} \left\langle \ln \left( \det \left( \mathbf{C} \hat{\boldsymbol{\Phi}} \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}}, \quad (17)$$

$$\text{subject to } \mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}, \quad (18)$$

where  $\hat{\boldsymbol{\Phi}} = \text{diag}(\phi(\hat{y}_1)^2, \dots, \phi(\hat{y}_{K_1})^2)$ ,  $\phi(\hat{y}_k) = a^{-1} |\partial g_k(\hat{y}_k) / \partial \hat{y}_k|$  ( $k = 1, \dots, K_1$ ),  $g_k(\hat{y}_k) = 2\sqrt{\tilde{f}(\hat{y}_k; \tilde{\boldsymbol{\theta}}_k)}$ ,  $\hat{y}_k = a^{-1} y_k = \mathbf{c}_k^T \hat{\mathbf{x}}$ , and  $\hat{\mathbf{x}} = \boldsymbol{\Sigma}_0^{-1/2} \mathbf{U}_0^T \mathbf{x}$ . We apply the gradient descent method to optimize the objective function, with the gradient of  $Q[\mathbf{C}]$  given by:

$$\frac{dQ[\mathbf{C}]}{d\mathbf{C}} = - \left\langle \left( \mathbf{C} \hat{\boldsymbol{\Phi}} \mathbf{C}^T \right)^{-1} \mathbf{C} \hat{\boldsymbol{\Phi}} + \hat{\mathbf{x}} \boldsymbol{\omega}^T \right\rangle_{\hat{\mathbf{x}}}, \quad (19)$$

where  $\boldsymbol{\omega} = (\omega_1, \dots, \omega_{K_1})^T$ ,  $\omega_k = \phi(\hat{y}_k) \phi'(\hat{y}_k) \mathbf{c}_k^T \left( \mathbf{C} \hat{\boldsymbol{\Phi}} \mathbf{C}^T \right)^{-1} \mathbf{c}_k$ ,  $k = 1, \dots, K_1$ . Usually, for optimizing the objective in Eq. 17, the orthogonality constraint (Eq. 18) is unnecessary. This orthogonality constraint can accelerate the convergence rate if we employ it for the initial iterations.

When  $K_0 = K_1$  (or  $K_0 > K_1$ ), the objective function  $Q[\mathbf{C}]$  can be reduced to a simpler form, and its gradient is also easy to compute (see Appendix A.4.1). However, when  $K_0 < K_1$ , it is computationally expensive to update  $\mathbf{C}$  by applying the gradient of  $Q[\mathbf{C}]$  directly, since it requires matrix inversion for every  $\hat{\mathbf{x}}$ . We use another objective function  $\hat{Q}[\mathbf{C}]$  (see Eq. A.118) which is an

approximation to  $Q[\mathbf{C}]$ , but its gradient is easier to compute (see Appendix A.4.2). The function  $\hat{Q}[\mathbf{C}]$  is the approximation of  $Q[\mathbf{C}]$ , ideally they have the same optimal solution for the parameter  $\mathbf{C}$ .

Usually, for solving the objective (17), the orthogonality constraint (Eq. 18) is not necessary. However, this orthogonality constraint can accelerate the convergence rate if we employ it for the initial iteration to update  $\mathbf{C}$  (see Appendix A.5).

### 3 EXPERIMENTAL RESULTS

We have applied our methods to the natural images from Olshausen’s image dataset (Olshausen & Field, 1996) and the images of handwritten digits from MNIST dataset (LeCun et al., 1998) using Matlab 2016a on a computer with 12 Intel CPU cores (2.4 GHz). The gray level of each raw image was normalized to the range of 0 to 1.  $M$  image patches with size  $w \times w = K$  for training were randomly sampled from the images. We used the Poisson neuron model with a modified sigmoidal tuning function  $\tilde{f}(y; \tilde{\theta}) = \frac{1}{4(1+\exp(-\beta y - b))^2}$ , with  $g(y) = 2\sqrt{\tilde{f}(y; \tilde{\theta})} = \frac{1}{1+\exp(-\beta y - b)}$ , where  $\tilde{\theta} = (\beta, b)^T$ . We obtained the initial values (see Appendix A.3.2):  $b_0 = 0$  and  $\beta_0 \approx 1.81\sqrt{K_1 K_0^{-1}}$ . For our experiments, we set  $\beta = 0.5\beta_0$  for iteration epoch  $t = 1, \dots, t_0$  and  $\beta = \beta_0$  for  $t = t_0 + 1, \dots, t_{\max}$ , where  $t_0 = 50$ .

Firstly, we tested the case of  $K = K_0 = K_1 = 144$  and randomly sampled  $M = 10^5$  image patches with size  $12 \times 12$  from the Olshausen’s natural images, assuming that  $N = 10^6$  neurons were divided into  $K_1 = 144$  classes and  $\epsilon = 1$  (see Eq. A.52 in Appendix). The input patches were preprocessed by the ZCA whitening filters (see Eq. A.68). To test our algorithms, we chose the batch size to be equal to the number of training samples  $M$ , although we could also choose a smaller batch size. We updated the matrix  $\mathbf{C}$  from a random start, and set parameters  $t_{\max} = 300$ ,  $v_1 = 0.4$ , and  $\tau = 0.8$  for all experiments.

In this case, the optimal solution  $\mathbf{C}$  looked similar to the optimal solution of IICA (Bell & Sejnowski, 1997). We also compared with the fast ICA algorithm (FICA) (Hyvärinen, 1999), which is faster than IICA. We also tested the restricted Boltzmann machine (RBM) (Hinton et al., 2006) for a unsupervised learning of representations, and found that it could not easily learn Gabor-like filters from Olshausen’s image dataset as trained by contrastive divergence. However, an improved method by adding a sparsity constraint on the output units, e.g., sparse RBM (SRBM) (Lee et al., 2008) or sparse autoencoder (Hinton, 2010), could attain Gabor-like filters from this dataset. Similar results with Gabor-like filters were also reproduced by the denoising autoencoders (Vincent et al., 2010), which method requires a careful choice of parameters, such as noise level, learning rate, and batch size.

In order to compare our methods, i.e. Algorithm 1 (Alg.1, see Appendix A.4.1) and Algorithm 2 (Alg.2, see Appendix A.4.2), with other methods, i.e. IICA, FICA and SRBM, we implemented these algorithms using the same initial weights and the same training data set (i.e.  $10^5$  image patches preprocessed by the ZCA whitening filters). To get a good result by IICA, we must carefully select the parameters; we set the batch size as 50, the initial learning rate as 0.01, and final learning rate as 0.0001, with an exponential decay with the epoch of iterations. IICA tends to have a faster convergence rate for a bigger batch size but it may become harder to escape local minima. For FICA, we chose the nonlinearity function  $f(u) = \log \cosh(u)$  as contrast function, and for SRBM, we set the sparseness control constant  $p$  as 0.01 and 0.03. The number of epoches for iterations was set to 300 for all algorithms. Figure 2 shows the filters learned by our methods and other methods. Each filter in Figure 2(a) corresponds to a column vector of matrix  $\tilde{\mathbf{C}}$  (see Eq. A.69), where each vector for display is normalized by  $\tilde{\mathbf{c}}_k \leftarrow \tilde{\mathbf{c}}_k / \max(|\tilde{c}_{1,k}|, \dots, |\tilde{c}_{K,k}|)$ ,  $k = 1, \dots, K_1$ . The results in Figures 2(a), 2(b) and 2(c) look very similar to one another, and slightly different from the results in Figure 2(d) and 2(e). There are no Gabor-like filters in Figure 2(f), which corresponds to SRBM with  $p = 0.03$ .

Figure 3 shows how the coefficient entropy (CFE) (see Eq. A.122) and the conditional entropy (CDE) (see Eq. A.125) varied with training time. We calculated CFE and CDE by sampling once every 10 epoches from a total of 300 epoches. These results show that our algorithms had a fast convergence rate towards stable solutions while having CFE and CDE values similar to the algorithm

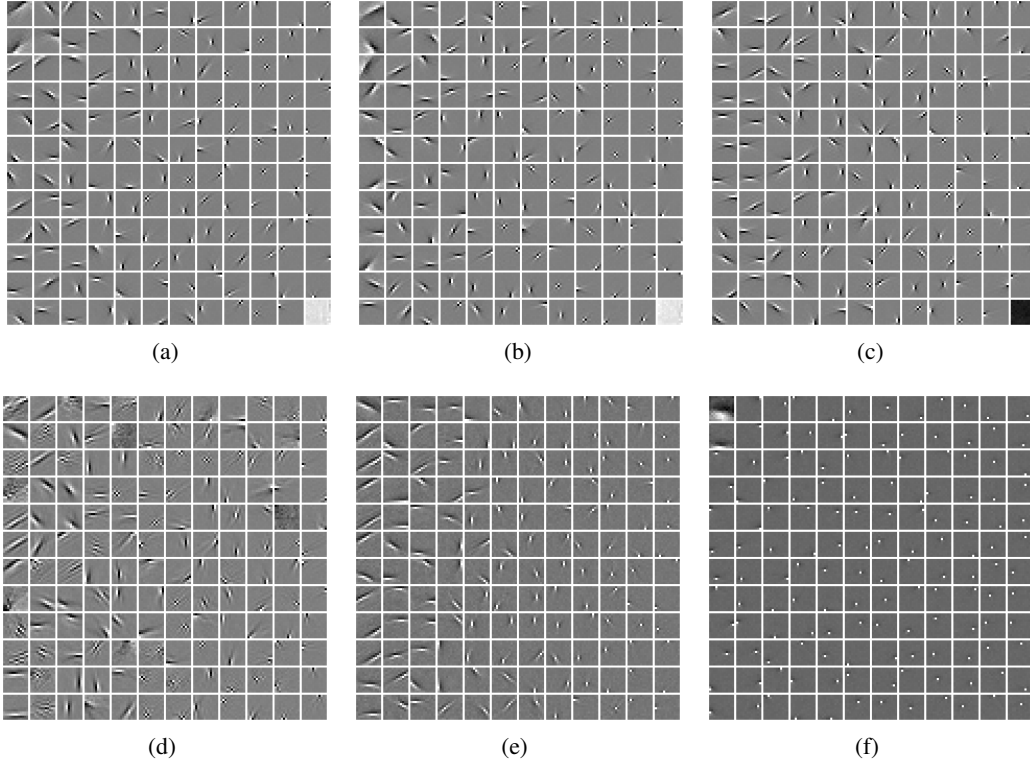


Figure 2: Comparison of filters obtained from  $10^5$  natural image patches of size  $12 \times 12$  by our methods (Alg.1 and Alg.2) and other methods. The number of output filters was  $K_1 = 144$ . (a): Alg.1. (b): Alg.2. (c): IICA. (d): FICA. (e): SRBM ( $p = 0.01$ ). (f): SRBM ( $p = 0.03$ ).

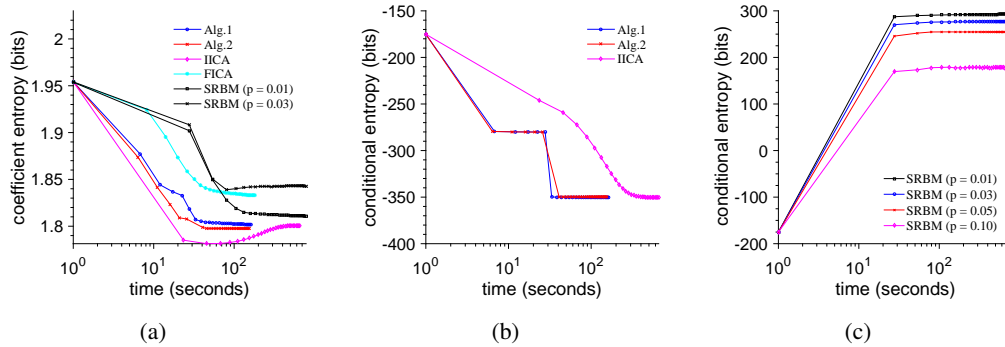


Figure 3: Comparison of quantization effects and convergence rate by coefficient entropy (see A.122) and conditional entropy (see A.125) corresponding to training results (filters) shown in Figure 2. The coefficient entropy (panel a) and conditional entropy (panel b and c) are shown as a function of training time on a logarithmic scale. All experiments run on the same machine using Matlab. Here we sampled once every 10 epoches out of a total of 300 epoches. We set epoch number  $t_0 = 50$  for Alg.1 and Alg.2 and the start time to 1 second.

of IICA, which converged much more slowly. Here the values of CFE and CDE should be as small as possible for a good representation learned from the same data set. Here we set epoch number  $t_0 = 50$  in our algorithms (see Alg.1 and Alg.2), and the start time was set to 1 second. This explains the step seen in Figure 3 (b) for Alg.1 and Alg.2 since the parameter  $\beta$  was updated when epoch number  $t = t_0$ . FICA had a convergence rate close to our algorithms but had a big CFE, which is reflected by the quality of the filter results in Figure 2. The convergence rate and CFE for SRBM were close to IICA, but SRBM had a much bigger CDE than IICA, which implies that the information had a greater loss when passing through the system optimized by SRBM than by IICA or our methods.

From Figure 3(c) we see that the CDE (or MI  $I(X; R)$ , see Eq. A.124 and A.125) decreases (or increases) with the increase of the value of the sparseness control constant  $p$ . Note that a smaller  $p$  means sparser outputs. Hence, in this sense, increasing sparsity may result in sacrificing some information. On the other hand, a weak sparsity constraint may lead to failure of learning Gabor-like filters (see Figure 2(f)), and increasing sparsity has an advantage in reducing the impact of noise in many practical cases. Similar situation also occurs in sparse coding (Olshausen & Field, 1997), which provides a class of algorithms for learning overcomplete dictionary representations of the input signals. However, its training is time consuming due to its expensive computational cost, although many new training algorithms have emerged (e.g. Aharon et al., 2006; Elad & Aharon, 2006; Lee et al., 2006; Mairal et al., 2010). See Appendix A.5 for additional experimental results.

## 4 CONCLUSIONS

In this paper, we have presented a framework for unsupervised learning of representations via information maximization for neural populations. Information theory is a powerful tool for machine learning and it also provides a benchmark of optimization principle for neural information processing in nervous systems. Our framework is based on an asymptotic approximation to MI for a large-scale neural population. To optimize the infomax objective, we first use hierarchical infomax to obtain a good approximation to the global optimal solution. Analytical solutions of the hierarchical infomax are further improved by a fast convergence algorithm based on gradient descent. This method allows us to optimize highly nonlinear neural networks via hierarchical optimization using infomax principle.

From the viewpoint of information theory, the unsupervised pre-training for deep learning (Hinton & Salakhutdinov, 2006; Bengio et al., 2007) may be reinterpreted as a process of hierarchical infomax, which might help explain why unsupervised pre-training helps deep learning (Erhan et al., 2010). In our framework, a pre-whitening step can emerge naturally by the hierarchical infomax, which might also explain why a pre-whitening step is useful for training in many learning algorithms (Coates et al., 2011; Bengio, 2012).

Our model naturally incorporates a considerable degree of biological realism. It allows the optimization of a large-scale neural population with noisy spiking neurons while taking into account of multiple biological constraints, such as membrane noise, limited energy, and bounded connection weights. We employ a technique to attain a low-rank weight matrix for optimization, so as to reduce the influence of noise and discourage overfitting during training. In our model, many parameters are optimized, including the population density of parameters, filter weight vectors, and parameters for nonlinear tuning functions. Optimizing all these model parameters could not be easily done by many other methods.

Our experimental results suggest that our method for unsupervised learning of representations has obvious advantages in its training speed and robustness over the main existing methods. Our model has a nonlinear feedforward structure and is convenient for fast learning and inference. This simple and flexible framework for unsupervised learning of presentations should be readily extended to training deep structure networks. In future work, it would be interesting to use our method to train deep structure networks with either unsupervised or supervised learning.

## ACKNOWLEDGMENTS

We thank Prof. Honglak Lee for sharing Matlab code for algorithm comparison, Prof. Shan Tan for discussions and comments and Kai Liu for helping draw Figure 1. Supported by grant NIH-NIDCD R01 DC013698.

## REFERENCES

Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11), 4311–4322.

- Amari, S. (1999). Natural gradient learning for over- and under-complete bases in ica. *Neural Comput.*, 11(8), 1875–1883.
- Atick, J. J. (1992). Could information theory provide an ecological theory of sensory processing? *Network: Comp. Neural.*, 3(2), 213–251.
- Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. *Sensory Communication*, (pp. 217–234).
- Bell, A. J. & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Comput.*, 7(6), 1129–1159.
- Bell, A. J. & Sejnowski, T. J. (1997). The “independent components” of natural scenes are edge filters. *Vision Res.*, 37(23), 3327–3338.
- Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. *Unsupervised and Transfer Learning Challenges in Machine Learning*, 7, 19.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8), 1798–1828.
- Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 153.
- Borst, A. & Theunissen, F. E. (1999). Information theory and neural coding. *Nature neuroscience*, 2(11), 947–957.
- Carlo, C. N. & Stevens, C. F. (2013). Structural uniformity of neocortex, revisited. *Proceedings of the National Academy of Sciences*, 110(4), 1488–1493.
- Coates, A., Ng, A. Y., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics* (pp. 215–223).
- Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.
- Cover, T. M. & Thomas, J. A. (2006). *Elements of Information, 2nd Edition*. New York: Wiley-Interscience.
- Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2), 303–353.
- Elad, M. & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12), 3736–3745.
- Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11, 625–660.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672–2680).
- Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum*, 9(1), 926.
- Hinton, G., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7), 1527–1554.
- Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.
- Huang, W. & Zhang, K. (2016). Information-theoretic bounds and approximations in neural population coding. *Neural Comput*, submitted, URL <https://arxiv.org/abs/1611.01414>.

- Hubel, D. H. & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1), 106–154.
- Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *Neural Networks, IEEE Transactions on*, 10(3), 626–634.
- Karklin, Y. & Simoncelli, E. P. (2011). Efficient coding of natural images with a population of noisy linear-nonlinear neurons. In *Advances in neural information processing systems*, volume 24 (pp. 999–1007).
- Konstantinides, K. & Yao, K. (1988). Statistical analysis of effective singular values in matrix rank determination. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(5), 757–763.
- Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T. S., & Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural computation*, 15(2), 349–396.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Lee, H., Battle, A., Raina, R., & Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in neural information processing systems* (pp. 801–808).
- Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems* (pp. 873–880).
- Lewicki, M. S. & Olshausen, B. A. (1999). Probabilistic framework for the adaptation and comparison of image codes. *JOSA A*, 16(7), 1587–1601.
- Lewicki, M. S. & Sejnowski, T. J. (2000). Learning overcomplete representations. *Neural computation*, 12(2), 337–365.
- Linsker, R. (1988). Self-Organization in a perceptual network. *Computer*, 21(3), 105–117.
- Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning* (pp. 689–696).: ACM.
- Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11, 19–60.
- Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems* (pp. 2924–2932).
- Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 807–814).
- Olshausen, B. A. & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 607–609.
- Olshausen, B. A. & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Res.*, 37(23), 3311–3325.
- Rao, C. R. (1945). Information and accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37(3), 81–91.
- Shannon, C. (1948). A mathematical theory of communications. *Bell System Technical Journal*, 27, 379–423 and 623–656.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11, 3371–3408.

Yarrow, S., Challis, E., & Series, P. (2012). Fisher and shannon information in finite neural populations. *Neural computation*, 24(7), 1740–1780.

## APPENDIX

### A.1 FORMULAS FOR APPROXIMATION OF MUTUAL INFORMATION

It follows from  $I(X; R) = \left\langle \ln \frac{p(\mathbf{x}|\mathbf{r})}{p(\mathbf{x})} \right\rangle_{\mathbf{r}, \mathbf{x}}$  and Eq. (1) that the conditional entropy should read:

$$H(X|R) = -\langle \ln p(\mathbf{x}|\mathbf{r}) \rangle_{\mathbf{r}, \mathbf{x}} \simeq -\frac{1}{2} \left\langle \ln \left( \det \left( \frac{\mathbf{G}(\mathbf{x})}{2\pi e} \right) \right) \right\rangle_{\mathbf{x}}. \quad (\text{A.1})$$

The Fisher information matrix  $\mathbf{J}(\mathbf{x})$  (see Eq. 3), which is symmetric and positive semidefinite, can be written also as

$$\mathbf{J}(\mathbf{x}) = \left\langle \frac{\partial \ln p(\mathbf{r}|\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \ln p(\mathbf{r}|\mathbf{x})}{\partial \mathbf{x}^T} \right\rangle_{\mathbf{r}|\mathbf{x}}. \quad (\text{A.2})$$

If we suppose  $p(\mathbf{r}|\mathbf{x})$  is conditional independent, namely,  $p(\mathbf{r}|\mathbf{x}) = \prod_{n=1}^N p(r_n|\mathbf{x}; \boldsymbol{\theta}_n)$ , then we have (see Huang & Zhang, 2016)

$$\mathbf{J}(\mathbf{x}) = N \int_{\Theta} p(\boldsymbol{\theta}) \mathbf{S}(\mathbf{x}; \boldsymbol{\theta}) d\boldsymbol{\theta}, \quad (\text{A.3})$$

$$\mathbf{S}(\mathbf{x}; \boldsymbol{\theta}) = \left\langle \frac{\partial \ln p(r|\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \ln p(r|\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}^T} \right\rangle_{r|\mathbf{x}}, \quad (\text{A.4})$$

where  $p(\boldsymbol{\theta})$  is the population density function of parameter  $\boldsymbol{\theta}$ ,

$$p(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_n), \quad (\text{A.5})$$

and  $\delta(\cdot)$  denotes the Dirac delta function. It can be proved that the approximation function of MI  $I_G[p(\boldsymbol{\theta})]$  (Eq. 1) is concave about  $p(\boldsymbol{\theta})$  (Huang & Zhang, 2016). In Eq. (A.3), we can approximate the continuous integral by a discrete summation for numerical computation,

$$\mathbf{J}(\mathbf{x}) \approx N \sum_{k=1}^{K_1} \alpha_k \mathbf{S}(\mathbf{x}; \boldsymbol{\theta}_k), \quad (\text{A.6})$$

where  $\sum_{k=1}^{K_1} \alpha_k = 1$ ,  $\alpha_k > 0$ ,  $k = 1, \dots, K_1$ ,  $1 \leq K_1 \leq N$ .

For Poisson neuron model, by Eq. (A.4) we have (see Huang & Zhang, 2016)

$$p(r|\mathbf{x}; \boldsymbol{\theta}) = \frac{f(\mathbf{x}; \boldsymbol{\theta})^r}{r!} \exp(-f(\mathbf{x}; \boldsymbol{\theta})), \quad (\text{A.7})$$

$$\begin{aligned} \mathbf{S}(\mathbf{x}; \boldsymbol{\theta}) &= \frac{1}{f(\mathbf{x}; \boldsymbol{\theta})} \frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}^T} \\ &= \frac{\partial g(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial g(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}^T}, \end{aligned} \quad (\text{A.8})$$

where  $f(\mathbf{x}; \boldsymbol{\theta}) \geq 0$  is the activation function (mean response) of neuron and

$$g(\mathbf{x}; \boldsymbol{\theta}) = 2\sqrt{f(\mathbf{x}; \boldsymbol{\theta})}. \quad (\text{A.9})$$

Similarly, for Gaussian noise model, we have

$$p(r|\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(r - f(\mathbf{x}; \boldsymbol{\theta}))^2}{2\sigma^2}\right), \quad (\text{A.10})$$

$$\mathbf{S}(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{\sigma^2} \frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \mathbf{x}^T}, \quad (\text{A.11})$$

where  $\sigma > 0$  denotes the standard deviation of noise.

Sometimes we do not know the specific form of  $p(\mathbf{x})$  and only know  $M$  samples,  $\mathbf{x}_1, \dots, \mathbf{x}_M$ , which are independent and identically distributed (i.i.d.) samples drawn from the distribution  $p(\mathbf{x})$ . Then we can use the empirical average to approximate the integral in Eq. (1):

$$I_G \approx \frac{1}{2} \sum_{m=1}^M \ln(\det(\mathbf{G}(\mathbf{x}_m))) + H(X). \quad (\text{A.12})$$

## A.2 PROOF OF PROPOSITION 1

**Proof.** It follows from the data-processing inequality (Cover & Thomas, 2006) that

$$I(X; R) \leq I(Y; R) \leq I(\check{Y}; R) \leq I(\bar{Y}; R), \quad (\text{A.13})$$

$$I(X; R) \leq I(X; \bar{Y}) \leq I(X; \check{Y}) \leq I(X; Y). \quad (\text{A.14})$$

Since

$$p(\bar{y}_k|\mathbf{x}) = p(\check{y}_{k_1}, \dots, \check{y}_{k_{N_k}}|\mathbf{x}) = \mathcal{N}(\mathbf{w}_k^T \mathbf{x}, N_k^{-1} \sigma^2), k = 1, \dots, K_1, \quad (\text{A.15})$$

we have

$$p(\bar{\mathbf{y}}|\mathbf{x}) = p(\check{\mathbf{y}}|\mathbf{x}), \quad (\text{A.16})$$

$$p(\bar{\mathbf{y}}) = p(\check{\mathbf{y}}), \quad (\text{A.17})$$

$$I(X; \bar{Y}) = I(X; \check{Y}). \quad (\text{A.18})$$

Hence, by (A.14) and (A.18), expression (12) holds.

On the other hand, when  $N_k$  is large, from Eq. (10) we know that the distribution of  $\bar{Z}_k$ , namely,  $\mathcal{N}(0, N_k^{-1} \sigma^2)$ , approaches a Dirac delta function  $\delta(\bar{z}_k)$ . Then by (7) and (9) we have  $p(\mathbf{r}|\bar{\mathbf{y}}) \simeq p(\mathbf{r}|\mathbf{y}) = p(\mathbf{r}|\mathbf{x})$  and

$$I(X; R) = I(Y; R) - \left\langle \ln \frac{p(\mathbf{r}|\mathbf{y})}{p(\mathbf{r}|\mathbf{x})} \right\rangle_{\mathbf{r}, \mathbf{x}} = I(Y; R), \quad (\text{A.19})$$

$$I(Y; R) = I(\bar{Y}; R) - \left\langle \ln \frac{p(\mathbf{r}|\bar{\mathbf{y}})}{p(\mathbf{r}|\mathbf{y})} \right\rangle_{\mathbf{r}, \mathbf{y}, \bar{\mathbf{y}}} \simeq I(\bar{Y}; R), \quad (\text{A.20})$$

$$I(Y; R) = I(\check{Y}; R) - \left\langle \ln \frac{p(\mathbf{r}|\check{\mathbf{y}})}{p(\mathbf{r}|\mathbf{y})} \right\rangle_{\mathbf{r}, \mathbf{y}, \check{\mathbf{y}}} \simeq I(\check{Y}; R), \quad (\text{A.21})$$

$$I(X; Y) = I(X; \bar{Y}) - \left\langle \ln \frac{p(\mathbf{x}|\bar{\mathbf{y}})}{p(\mathbf{x}|\mathbf{y})} \right\rangle_{\mathbf{x}, \mathbf{y}, \bar{\mathbf{y}}} \simeq I(X; \bar{Y}). \quad (\text{A.22})$$

It follows from (A.13) and (A.19) that (11) holds. Combining (11), (12) and (A.20)–(A.22), we immediately get (13) and (14). This completes the proof of **Proposition 1**.  $\square$

## A.3 HIERARCHICAL OPTIMIZATION FOR MAXIMIZING $I(X; R)$

In the following, we will discuss the optimization procedure for maximizing  $I(X; R)$  in two stages: maximizing  $I(X; \check{Y})$  and maximizing  $I(Y; R)$ .



### A.3.1 THE 1ST STAGE

In the first stage, our goal is to maximize the MI  $I(X; \check{Y})$  and get the optimal parameters  $\mathbf{w}_k$  ( $k = 1, \dots, K_1$ ). Assume that the stimulus  $\mathbf{x}$  has zero mean (if not, let  $\mathbf{x} \leftarrow \mathbf{x} - \langle \mathbf{x} \rangle_{\mathbf{x}}$ ) and covariance matrix  $\Sigma_{\mathbf{x}}$ . It follows from eigendecomposition that

$$\Sigma_{\mathbf{x}} = \langle \mathbf{x}\mathbf{x}^T \rangle_{\mathbf{x}} \approx \frac{1}{M-1} \mathbf{X}\mathbf{X}^T = \mathbf{U}\Sigma\mathbf{U}^T, \quad (\text{A.23})$$

where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$ ,  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K] \in \mathbb{R}^{K \times K}$  is an unitary orthogonal matrix and  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_K^2)$  is a positive diagonal matrix with  $\sigma_1 \geq \dots \geq \sigma_K > 0$ . Define

$$\tilde{\mathbf{x}} = \Sigma^{-1/2} \mathbf{U}^T \mathbf{x}, \quad (\text{A.24})$$

$$\tilde{\mathbf{w}}_k = \Sigma^{1/2} \mathbf{U}^T \mathbf{w}_k, \quad (\text{A.25})$$

$$y_k = \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}, \quad (\text{A.26})$$

where  $k = 1, \dots, K_1$ . The covariance matrix of  $\tilde{\mathbf{x}}$  is given by

$$\Sigma_{\tilde{\mathbf{x}}} = \langle \tilde{\mathbf{x}}\tilde{\mathbf{x}}^T \rangle_{\tilde{\mathbf{x}}} \approx \mathbf{I}_K, \quad (\text{A.27})$$

and  $\mathbf{I}_K$  is a  $K \times K$  identity matrix. From (1) and (A.11) we have  $I(X; \check{Y}) = I(\tilde{X}; \check{Y})$  and

$$I(\tilde{X}; \check{Y}) \simeq I'_G = \frac{1}{2} \ln \left( \det \left( \frac{\tilde{\mathbf{G}}}{2\pi e} \right) \right) + H(\tilde{X}), \quad (\text{A.28})$$

$$\tilde{\mathbf{G}} \approx N\sigma^{-2} \sum_{k=1}^{K_1} \alpha_k \tilde{\mathbf{w}}_k \tilde{\mathbf{w}}_k^T + \mathbf{I}_K. \quad (\text{A.29})$$

The following approximations are useful (see Huang & Zhang, 2016):

$$p(\tilde{\mathbf{x}}) \approx \mathcal{N}(0, \mathbf{I}_K), \quad (\text{A.30})$$

$$\mathbf{P}(\tilde{\mathbf{x}}) = -\frac{\partial^2 \ln p(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}} \partial \tilde{\mathbf{x}}^T} \approx \mathbf{I}_K. \quad (\text{A.31})$$

By the central limit theorem, the distribution of random variable  $\tilde{X}$  is closer to a normal distribution than the distribution of the original random variable  $X$ . On the other hand, the PCA models assume multivariate gaussian data whereas the ICA models assume multivariate non-gaussian data. Hence by a PCA-like whitening transformation (A.24) we can use the approximation (A.31) with the Laplace's method of asymptotic expansion, which only requires that the peak be close to its mean while random variable  $\tilde{X}$  needs not be exactly Gaussian.

Without any constraints on the Gaussian channel of neural populations, especially the peak firing rates, the capacity of this channel may grow indefinitely:  $I(\tilde{X}; \check{Y}) \rightarrow \infty$ . The most common constraint on the neural populations is an energy or power constraint which can also be regarded as a signal-to-noise ratio (SNR) constraint. The SNR for the output  $\check{y}_n$  of the  $n$ -th neuron is given by

$$\text{SNR}_n = \frac{1}{\sigma^2} \langle (\mathbf{w}_n^T \mathbf{x})^2 \rangle_{\mathbf{x}} \approx \frac{1}{\sigma^2} \tilde{\mathbf{w}}_n^T \tilde{\mathbf{w}}_n, n = 1, \dots, N. \quad (\text{A.32})$$

We require that

$$\frac{1}{N} \sum_{n=1}^N \text{SNR}_n \approx \frac{1}{\sigma^2} \sum_{k=1}^{K_1} \alpha_k \tilde{\mathbf{w}}_k \tilde{\mathbf{w}}_k^T \leq \rho, \quad (\text{A.33})$$

where  $\rho$  is a positive constant. Then by Eq. (A.28), (A.29) and (A.33), we have the following optimization problem:

$$\text{minimize } Q'_G[\hat{\mathbf{W}}] = -\frac{1}{2} \ln \left( \det \left( N\sigma^{-2} \hat{\mathbf{W}} \hat{\mathbf{W}}^T + \mathbf{I}_K \right) \right), \quad (\text{A.34})$$

$$\text{subject to } h = \text{Tr} \left( \hat{\mathbf{W}} \hat{\mathbf{W}}^T \right) - E \leq 0, \quad (\text{A.35})$$

where  $\text{Tr}(\cdot)$  denotes matrix trace and

$$\hat{\mathbf{W}} = \tilde{\mathbf{W}} \mathbf{A}^{1/2} = \Sigma^{1/2} \mathbf{U}^T \mathbf{W} \mathbf{A}^{1/2} = [\hat{\mathbf{w}}_1, \dots, \hat{\mathbf{w}}_{K_1}], \quad (\text{A.36})$$

$$\mathbf{A} = \text{diag}(\alpha_1, \dots, \alpha_{K_1}), \quad (\text{A.37})$$

$$\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_{K_1}], \quad (\text{A.38})$$

$$\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_{K_1}], \quad (\text{A.39})$$

$$E = \rho \sigma^2. \quad (\text{A.40})$$

Here  $E$  is a constant that does not affect the final optimal solution so we set  $E = 1$ . Then we obtain an optimal solution as follows:

$$\mathbf{W} = a \mathbf{U}_0 \Sigma_0^{-1/2} \mathbf{V}_0^T, \quad (\text{A.41})$$

$$\mathbf{A} = K_1^{-1} \mathbf{I}_{K_1}, \quad (\text{A.42})$$

$$a = \sqrt{E K_1 K_0^{-1}} = \sqrt{K_1 K_0^{-1}}, \quad (\text{A.43})$$

$$\Sigma_0 = \text{diag}(\sigma_1^2, \dots, \sigma_{K_0}^2), \quad (\text{A.44})$$

$$\mathbf{U}_0 = \mathbf{U}(:, 1:K_0) \in \mathbb{R}^{K \times K_0}, \quad (\text{A.45})$$

$$\mathbf{V}_0 = \mathbf{V}(:, 1:K_0) \in \mathbb{R}^{K_1 \times K_0}, \quad (\text{A.46})$$

where  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{K_1}]$  is an  $K_1 \times K_1$  unitary orthogonal matrix, parameter  $K_0$  represents the size of the reduced dimension ( $1 \leq K_0 \leq K$ ), and its value will be determined below. Now the optimal parameters  $\mathbf{w}_n$  ( $n = 1, \dots, N$ ) are clustered into  $K_1$  classes (see Eq. A.6) and obey an uniform discrete distribution (see also Eq. A.60 in Appendix A.3.2).

When  $K = K_0 = K_1$ , the optimal solution of  $\mathbf{W}$  in Eq. (A.41) is a whitening-like filter. When  $\mathbf{V} = \mathbf{I}_K$ , the optimal matrix  $\mathbf{W}$  is the principal component analysis (PCA) whitening filters. In the symmetrical case with  $\mathbf{V} = \mathbf{U}$ , the optimal matrix  $\mathbf{W}$  becomes a zero component analysis (ZCA) whitening filter. If  $K < K_1$ , this case leads to an overcomplete solution, whereas when  $K_0 \leq K_1 < K$ , the undercomplete solution arises. Since  $K_0 \leq K_1$  and  $K_0 \leq K$ ,  $Q'_G$  achieves its minimum when  $K_0 = K$ . However, in practice other factors may prevent it from reaching this minimum. For example, consider the average of squared weights,

$$\varsigma = \sum_{k=1}^{K_1} \alpha_k \|\mathbf{w}_k\|^2 = \text{Tr}(\mathbf{W} \mathbf{A} \mathbf{W}^T) = \frac{E}{K_0} \sum_{k=1}^{K_0} \sigma_k^{-2}, \quad (\text{A.47})$$

where  $\|\cdot\|$  denotes the Frobenius norm. The value of  $\varsigma$  is extremely large when any  $\sigma_k$  becomes vanishingly small. For real neurons these weights of connection are not allowed to be too large. Hence we impose a limitation on the weights:  $\varsigma \leq E_1$ , where  $E_1$  is a positive constant. This yields another constraint on the objective function,

$$\tilde{h} = \frac{E}{K_0} \sum_{k=1}^{K_0} \sigma_k^{-2} - E_1 \leq 0. \quad (\text{A.48})$$

From (A.35) and (A.48) we get the optimal  $K_0 = \arg \max_{\tilde{K}_0} \left( E \tilde{K}_0^{-1} \sum_{k=1}^{\tilde{K}_0} \sigma_k^{-2} \right)$ . By this constraint, small values of  $\sigma_k^2$  will often result in  $K_0 < K$  and a low-rank matrix  $\mathbf{W}$  (Eq. A.41).

On the other hand, the low-rank matrix  $\mathbf{W}$  can filter out the noise of stimulus  $\mathbf{x}$ . Consider the transformation  $\mathbf{Y} = \mathbf{W}^T \mathbf{X}$  with  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$  and  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_M]$  for  $M$  samples. It follows from the singular value decomposition (SVD) of  $\mathbf{X}$  that

$$\mathbf{X} = \mathbf{U} \mathbf{S} \tilde{\mathbf{V}}^T, \quad (\text{A.49})$$

where  $\mathbf{U}$  is given in (A.23),  $\tilde{\mathbf{V}}$  is a  $M \times M$  unitary orthogonal matrix,  $\mathbf{S}$  is a  $K \times M$  diagonal matrix with non-negative real numbers on the diagonal,  $S_{k,k} = \sqrt{M-1} \sigma_k$  ( $k = 1, \dots, K$ ,  $K \leq M$ ), and  $\mathbf{S} \mathbf{S}^T = (M-1) \Sigma$ . Let

$$\check{\mathbf{X}} = \sqrt{M-1} \mathbf{U}_0 \Sigma_0^{1/2} \tilde{\mathbf{V}}_0^T \approx \mathbf{X}, \quad (\text{A.50})$$

where  $\tilde{\mathbf{V}}_0 = \tilde{\mathbf{V}}(:, 1:K_0) \in \mathbb{R}^{M \times K_0}$ ,  $\Sigma_0$  and  $\mathbf{U}_0$  are given in (A.44) and (A.45), respectively. Then

$$\mathbf{Y} = \mathbf{W}^T \mathbf{X} = a \mathbf{V}_0 \Sigma_0^{-1/2} \mathbf{U}_0^T \mathbf{U} \mathbf{S} \tilde{\mathbf{V}}^T = \mathbf{W}^T \check{\mathbf{X}} = a \sqrt{M-1} \mathbf{V}_0 \tilde{\mathbf{V}}_0^T, \quad (\text{A.51})$$

where  $\check{\mathbf{X}}$  can be regarded as a denoised version of  $\mathbf{X}$ . The determination of the effective rank  $K_0 \leq K$  of the matrix  $\check{\mathbf{X}}$  by using singular values is based on various criteria (Konstantinides & Yao, 1988). Here we choose  $K_0$  as follows:

$$K_0 = \arg \min_{K'_0} \left( \sqrt{\frac{\sum_{k=1}^{K'_0} \sigma_k^2}{\sum_{k=1}^K \sigma_k^2}} \geq \epsilon \right), \quad (\text{A.52})$$

where  $\epsilon$  is a positive constant ( $0 < \epsilon \leq 1$ ).

Another advantage of a low-rank matrix  $\mathbf{W}$  is that it can significantly reduce overfitting for learning neural population parameters. In practice, the constraint (A.47) is equivalent to a weight-decay regularization term used in many other optimization problems (Cortes & Vapnik, 1995; Hinton, 2010), which can reduce overfitting to the training data. To prevent the neural networks from overfitting, Srivastava et al. (2014) presented a technique to randomly drop units from the neural network during training, which may in fact be regarded as an attempt to reduce the rank of the weight matrix because the dropout can result in a sparser weights (lower rank matrix). This means that the update is only concerned with keeping the more important components, which is similar to first performing a denoising process by the SVD low rank approximation.

In this stage, we have obtained the optimal parameter  $\mathbf{W}$  (see A.41). The optimal value of matrix  $\mathbf{V}_0$  can also be determined, as shown in Appendix A.3.3.

### A.3.2 THE 2ND STAGE

For this stage, our goal is to maximize the MI  $I(Y; R)$  and get the optimal parameters  $\tilde{\boldsymbol{\theta}}_k$ ,  $k = 1, \dots, K_1$ . Here the input is  $\mathbf{y} = (y_1, \dots, y_{K_1})^T$  and the output  $\mathbf{r} = (r_1, \dots, r_N)^T$  is also clustered into  $K_1$  classes. The responses of  $N_k$  neurons in the  $k$ -th subpopulation obey a Poisson distribution with mean  $\tilde{f}(\mathbf{e}_k^T \mathbf{y}; \tilde{\boldsymbol{\theta}}_k)$ , where  $\mathbf{e}_k$  is a unit vector with 1 in the  $k$ -th element and  $y_k = \mathbf{e}_k^T \mathbf{y}$ . By (A.24) and (A.26), we have

$$\langle y_k \rangle_{y_k} = 0, \quad (\text{A.53})$$

$$\sigma_{y_k}^2 = \langle y_k^2 \rangle_{y_k} = \|\tilde{\mathbf{w}}_k\|^2. \quad (\text{A.54})$$

Then for large  $N$ , by (1)–(4) and (A.30) we can use the following approximation,

$$I(Y; R) \simeq \check{I}_F = \frac{1}{2} \left\langle \ln \left( \det \left( \frac{\check{\mathbf{J}}(\mathbf{y})}{2\pi e} \right) \right) \right\rangle_{\mathbf{y}} + H(Y), \quad (\text{A.55})$$

where

$$\check{\mathbf{J}}(\mathbf{y}) = \text{diag} \left( N\alpha_1 |g'_1(y_1)|^2, \dots, N\alpha_{K_1} |g'_{K_1}(y_{K_1})|^2 \right), \quad (\text{A.56})$$

$$g'_k(y_k) = \frac{\partial g_k(y_k)}{\partial y_k}, \quad k = 1, \dots, K_1, \quad (\text{A.57})$$

$$g_k(y_k) = 2\sqrt{\tilde{f}(y_k; \tilde{\boldsymbol{\theta}}_k)}, \quad k = 1, \dots, K_1. \quad (\text{A.58})$$

It is easy to get that

$$\begin{aligned} \check{I}_F &= \frac{1}{2} \sum_{k=1}^{K_1} \left\langle \ln \left( \frac{N\alpha_k |g'_k(y_k)|^2}{2\pi e} \right) \right\rangle_{\mathbf{y}} + H(Y) \\ &\leq \frac{1}{2} \sum_{k=1}^{K_1} \left\langle \ln \left( \frac{|g'_k(y_k)|^2}{2\pi e} \right) \right\rangle_{\mathbf{y}} - \frac{K_1}{2} \ln \left( \frac{K_1}{N} \right) + H(Y), \end{aligned} \quad (\text{A.59})$$

where the equality holds if and only if

$$\alpha_k = \frac{1}{K_1}, k = 1, \dots, K_1, \quad (\text{A.60})$$

which is consistent with Eq. (A.42).

On the other hand, it follows from the Jensen's inequality that

$$\begin{aligned} \tilde{I}_F &= \left\langle \ln \left( p(\mathbf{y})^{-1} \det \left( \frac{\check{\mathbf{J}}(\mathbf{y})}{2\pi e} \right)^{1/2} \right) \right\rangle_{\mathbf{y}} \\ &\leq \ln \int \det \left( \frac{\check{\mathbf{J}}(\mathbf{y})}{2\pi e} \right)^{1/2} d\mathbf{y}, \end{aligned} \quad (\text{A.61})$$

where the equality holds if and only if  $p(\mathbf{y})^{-1} \det \left( \check{\mathbf{J}}(\mathbf{y}) \right)^{1/2}$  is a constant, which implies that

$$p(\mathbf{y}) = \frac{\det \left( \check{\mathbf{J}}(\mathbf{y}) \right)^{1/2}}{\int \det \left( \check{\mathbf{J}}(\mathbf{y}) \right)^{1/2} d\mathbf{y}} = \frac{\prod_{k=1}^{K_1} |g'_k(y_k)|}{\int \prod_{k=1}^{K_1} |g'_k(y_k)| d\mathbf{y}}. \quad (\text{A.62})$$

From (A.61) and (A.62), maximizing  $\tilde{I}_F$  yields

$$p(y_k) = \frac{|g'_k(y_k)|}{\int |g'_k(y_k)| dy_k}, k = 1, \dots, K_1. \quad (\text{A.63})$$

We assume that (A.63) holds, at least approximately. Hence we can let the peak of  $g'_k(y_k)$  be at  $y_k = \langle y_k \rangle_{y_k} = 0$  and  $\langle y_k^2 \rangle_{y_k} = \sigma_{y_k}^2 = \|\tilde{\mathbf{w}}_k\|^2$ . Then combining (A.57), (A.61) and (A.63) we find the optimal parameters  $\tilde{\boldsymbol{\theta}}_k$  for the nonlinear functions  $\tilde{f}(y_k; \tilde{\boldsymbol{\theta}}_k)$ ,  $k = 1, \dots, K_1$ .

### A.3.3 THE FINAL OBJECTIVE FUNCTION

In the preceding sections we have obtained the initial optimal solutions by maximizing  $I(X; \check{Y})$  and  $I(Y; R)$ . In this section, we will discuss how to find the final optimal  $\mathbf{V}_0$  and other parameters by maximizing  $I(X; R)$  from the initial optimal solutions.

First, we have

$$\mathbf{y} = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = a\hat{\mathbf{y}}, \quad (\text{A.64})$$

where  $a$  is given in (A.43) and

$$\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_{K_1})^T = \mathbf{C}^T \hat{\mathbf{x}} = \check{\mathbf{C}}^T \tilde{\mathbf{x}}, \quad (\text{A.65})$$

$$\hat{\mathbf{x}} = \boldsymbol{\Sigma}_0^{-1/2} \mathbf{U}_0^T \mathbf{x}, \quad (\text{A.66})$$

$$\mathbf{C} = \mathbf{V}_0^T \in \mathbb{R}^{K_0 \times K_1}, \quad (\text{A.67})$$

$$\tilde{\mathbf{x}} = \mathbf{U}_0 \boldsymbol{\Sigma}_0^{-1/2} \mathbf{U}_0^T \mathbf{x} = \mathbf{U}_0 \hat{\mathbf{x}}, \quad (\text{A.68})$$

$$\check{\mathbf{C}} = \mathbf{U}_0 \mathbf{C} = [\check{c}_1, \dots, \check{c}_{K_1}]. \quad (\text{A.69})$$

It follows that

$$I(X; R) = I(\tilde{X}; R) \simeq \tilde{I}_G = \frac{1}{2} \left\langle \ln \left( \det \left( \frac{\mathbf{G}(\hat{\mathbf{x}})}{2\pi e} \right) \right) \right\rangle_{\hat{\mathbf{x}}} + H(\tilde{X}), \quad (\text{A.70})$$

$$\mathbf{G}(\hat{\mathbf{x}}) = N \hat{\mathbf{W}} \hat{\boldsymbol{\Phi}} \hat{\mathbf{W}}^T + \mathbf{I}_K, \quad (\text{A.71})$$

$$\hat{\mathbf{W}} = \boldsymbol{\Sigma}^{1/2} \mathbf{U}^T \mathbf{W} \mathbf{A}^{1/2} = a \sqrt{K_1^{-1} \mathbf{I}_{K_0}^K} \mathbf{C} = \sqrt{K_0^{-1}} \mathbf{I}_{K_0}^K \mathbf{C}, \quad (\text{A.72})$$

where  $\mathbf{I}_{K_0}^K$  is a  $K \times K_0$  diagonal matrix with value 1 on the diagonal and

$$\hat{\Phi} = \Phi^2, \quad (\text{A.73})$$

$$\Phi = \text{diag}(\phi(\hat{y}_1), \dots, \phi(\hat{y}_{K_1})), \quad (\text{A.74})$$

$$\phi(\hat{y}_k) = a^{-1} \left| \frac{\partial g_k(\hat{y}_k)}{\partial \hat{y}_k} \right|, \quad (\text{A.75})$$

$$g_k(\hat{y}_k) = 2\sqrt{\tilde{f}(\hat{y}_k; \tilde{\theta}_k)}, \quad (\text{A.76})$$

$$\hat{y}_k = a^{-1} y_k = \mathbf{c}_k^T \hat{\mathbf{x}}, k = 1, \dots, K_1. \quad (\text{A.77})$$

Then we have

$$\det(\mathbf{G}(\hat{\mathbf{x}})) = \det(NK_0^{-1} \mathbf{C} \hat{\Phi} \mathbf{C}^T + \mathbf{I}_{K_0}). \quad (\text{A.78})$$

For large  $N$  and  $K_0/N \rightarrow 0$ , we have

$$\det(\mathbf{G}(\hat{\mathbf{x}})) \approx \det(\mathbf{J}(\hat{\mathbf{x}})) = \det(NK_0^{-1} \mathbf{C} \hat{\Phi} \mathbf{C}^T), \quad (\text{A.79})$$

$$\tilde{I}_G \approx \tilde{I}_F = -Q - \frac{K}{2} \ln(2\pi e) - \frac{K_0}{2} \ln(\varepsilon) + H(\tilde{X}), \quad (\text{A.80})$$

$$Q = -\frac{1}{2} \left\langle \ln \left( \det(\mathbf{C} \hat{\Phi} \mathbf{C}^T) \right) \right\rangle_{\hat{\mathbf{x}}}, \quad (\text{A.81})$$

$$\varepsilon = \frac{K_0}{N}. \quad (\text{A.82})$$

Hence we can state the optimization problem as:

$$\text{minimize } Q[\mathbf{C}] = -\frac{1}{2} \left\langle \ln \left( \det(\mathbf{C} \hat{\Phi} \mathbf{C}^T) \right) \right\rangle_{\hat{\mathbf{x}}}, \quad (\text{A.83})$$

$$\text{subject to } \mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}. \quad (\text{A.84})$$

The gradient from (A.83) is given by:

$$\frac{dQ[\mathbf{C}]}{d\mathbf{C}} = - \left\langle \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right)^{-1} \mathbf{C} \hat{\Phi} + \hat{\mathbf{x}} \omega^T \right\rangle_{\hat{\mathbf{x}}}, \quad (\text{A.85})$$

where  $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_{K_1}]$ ,  $\omega = (\omega_1, \dots, \omega_{K_1})^T$ , and

$$\omega_k = \phi(\hat{y}_k) \phi'(\hat{y}_k) \mathbf{c}_k^T \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right)^{-1} \mathbf{c}_k, k = 1, \dots, K_1. \quad (\text{A.86})$$

In the following we will discuss how to get the optimal solution of  $\mathbf{C}$  for two specific cases.

#### A.4 ALGORITHMS FOR OPTIMIZATION OBJECTIVE FUNCTION

##### A.4.1 ALGORITHM 1: $K_0 = K_1$

Now  $\mathbf{C} \mathbf{C}^T = \mathbf{C}^T \mathbf{C} = \mathbf{I}_{K_1}$ , then by Eq. (A.83) we have

$$Q_1[\mathbf{C}] = - \left\langle \sum_{k=1}^{K_1} \ln(\phi(\hat{y}_k)) \right\rangle_{\hat{\mathbf{x}}}, \quad (\text{A.87})$$

$$\frac{dQ_1[\mathbf{C}]}{d\mathbf{C}} = - \langle \hat{\mathbf{x}} \omega^T \rangle_{\hat{\mathbf{x}}}, \quad (\text{A.88})$$

$$\omega_k = \frac{\phi'(\hat{y}_k)}{\phi(\hat{y}_k)}, k = 1, \dots, K_1. \quad (\text{A.89})$$

Under the orthogonality constraints (Eq. A.84), we can use the following update rule for learning  $\mathbf{C}$  (Edelman et al., 1998; Amari, 1999):

$$\mathbf{C}^{t+1} = \mathbf{C}^t + \mu_t \frac{d\mathbf{C}^t}{dt}, \quad (\text{A.90})$$

$$\frac{d\mathbf{C}^t}{dt} = - \frac{dQ_1[\mathbf{C}^t]}{d\mathbf{C}^t} + \mathbf{C}^t \left( \frac{dQ_1[\mathbf{C}^t]}{d\mathbf{C}^t} \right)^T \mathbf{C}^t, \quad (\text{A.91})$$

where the learning rate parameter  $\mu_t$  changes with the iteration count  $t$ ,  $t = 1, \dots, t_{\max}$ . Here we can use the empirical average to approximate the integral in (A.88) (see Eq. A.12). We can also apply stochastic gradient descent (SGD) method for online updating of  $\mathbf{C}^{t+1}$  in (A.90).

The orthogonality constraint (Eq. A.84) can accelerate the convergence rate. In practice, the orthogonality constraint (A.84) for objective function (A.83) is not strictly necessary in this case. We can completely discard this constraint condition and consider

$$\text{minimize } Q_2[\mathbf{C}] = - \left\langle \sum_{k=1}^{K_1} \ln(\phi(\hat{y}_k)) \right\rangle_{\hat{\mathbf{x}}} - \frac{1}{2} \ln(\det(\mathbf{C}^T \mathbf{C})), \quad (\text{A.92})$$

where we assume  $\text{rank}(\mathbf{C}) = K_1 = K_0$ . If we let

$$\frac{d\mathbf{C}}{dt} = -\mathbf{C}\mathbf{C}^T \frac{dQ_2[\mathbf{C}]}{d\mathbf{C}}, \quad (\text{A.93})$$

then

$$\text{Tr} \left( \frac{dQ_2[\mathbf{C}]}{d\mathbf{C}} \frac{d\mathbf{C}^T}{dt} \right) = -\text{Tr} \left( \mathbf{C}^T \frac{dQ_2[\mathbf{C}]}{d\mathbf{C}} \frac{dQ_2[\mathbf{C}]}{d\mathbf{C}^T} \mathbf{C} \right) \leq 0. \quad (\text{A.94})$$

Therefore we can use an update rule similar to Eq. A.90 for learning  $\mathbf{C}$ . In fact, the method can also be extended to the case  $K_0 > K_1$  by using the same objective function (A.92).

The learning rate parameter  $\mu_t$  (see A.90) is updated adaptively, as follows. First, calculate

$$\mu_t = \frac{v_t}{\kappa_t}, t = 1, \dots, t_{\max}, \quad (\text{A.95})$$

$$\kappa_t = \frac{1}{K_1} \sum_{k=1}^{K_1} \frac{\|\nabla \mathbf{C}^t(:, k)\|}{\|\mathbf{C}^t(:, k)\|}, \quad (\text{A.96})$$

and  $\mathbf{C}^{t+1}$  by (A.90) and (A.91), then calculate the value  $Q_1[\mathbf{C}^{t+1}]$ . If  $Q_1[\mathbf{C}^{t+1}] < Q_1[\mathbf{C}^t]$ , then let  $v_{t+1} \leftarrow v_t$ , continue for the next iteration; otherwise, let  $v_t \leftarrow \tau v_t$ ,  $\mu_t \leftarrow v_t / \kappa_t$  and recalculate  $\mathbf{C}^{t+1}$  and  $Q_1[\mathbf{C}^{t+1}]$ . Here  $0 < v_1 < 1$  and  $0 < \tau < 1$  are set as constants. After getting  $\mathbf{C}^{t+1}$  for each update, we employ a Gram–Schmidt orthonormalization process for matrix  $\mathbf{C}^{t+1}$ , where the orthonormalization process can accelerate the convergence. However, we can discard the Gram–Schmidt orthonormalization process after iterative  $t_0$  ( $> 1$ ) epochs for more accurate optimization solution  $\mathbf{C}$ . In this case, the objective function is given by the Eq. (A.92). We can also further optimize parameter  $b$  by gradient descent.

When  $K_0 = K_1$ , the objective function  $Q_2[\mathbf{C}]$  in Eq. (A.92) without constraint is the same as the objective function of infomax ICA (IICA) (Bell & Sejnowski, 1995; 1997), and as a consequence we should get the same optimal solution  $\mathbf{C}$ . Hence, in this sense, the IICA may be regarded as a special case of our method. Our method has a wider range of applications and can handle more generic situations. Our model is derived by neural populations with a huge number of neurons and it is not restricted to additive noise model. Moreover, our method has a faster convergence rate during training than IICA (see Section 3).

#### A.4.2 ALGORITHM 2: $K_0 \leq K_1$

In this case, it is computationally expensive to update  $\mathbf{C}$  by using the gradient of  $Q$  (see Eq. A.85), since it needs to compute the inverse matrix for every  $\hat{\mathbf{x}}$ . Here we provide an alternative method for learning the optimal  $\mathbf{C}$ . First, we consider the following inequalities.

**Proposition 2.** *The following inequations hold,*

$$\frac{1}{2} \left\langle \ln \left( \det \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right), \quad (\text{A.97})$$

$$\left\langle \ln \left( \det \left( \mathbf{C} \Phi \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq \ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right) \quad (\text{A.98})$$

$$\leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}}^2 \mathbf{C}^T \right) \right) \quad (\text{A.99})$$

$$\leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right), \quad (\text{A.100})$$

$$\ln \left( \det \left( \mathbf{C} \Phi \mathbf{C}^T \right) \right) \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right) \right), \quad (\text{A.101})$$

where  $\mathbf{C} \in \mathbb{R}^{K_0 \times K_1}$ ,  $K_0 \leq K_1$ , and  $\mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}$ .

**Proof.** Functions  $\ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right)$  and  $\ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right)$  are concave functions about  $p(\hat{\mathbf{x}})$  (see the proof of Proposition 5.2. in Huang & Zhang, 2016), which fact establishes inequalities (A.97) and (A.98).

Next we will prove the inequality (A.101). By SVD, we have

$$\mathbf{C} \Phi = \ddot{\mathbf{U}} \ddot{\mathbf{D}} \ddot{\mathbf{V}}^T, \quad (\text{A.102})$$

where  $\ddot{\mathbf{U}}$  is a  $K_0 \times K_0$  unitary orthogonal matrix,  $\ddot{\mathbf{V}} = [\ddot{\mathbf{v}}_1, \ddot{\mathbf{v}}_2, \dots, \ddot{\mathbf{v}}_{K_1}]$  is an  $K_1 \times K_1$  unitary orthogonal matrix, and  $\ddot{\mathbf{D}}$  is an  $K_0 \times K_1$  rectangular diagonal matrix with  $K_0$  positive real numbers on the diagonal. By the matrix Hadamard's inequality and Cauchy-Schwarz inequality we have

$$\begin{aligned} & \det \left( \mathbf{C} \Phi \mathbf{C}^T \mathbf{C} \Phi \mathbf{C}^T \right) \det \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right)^{-1} \\ &= \det \left( \ddot{\mathbf{D}} \ddot{\mathbf{V}}^T \mathbf{C}^T \mathbf{C} \ddot{\mathbf{V}} \ddot{\mathbf{D}}^T \left( \ddot{\mathbf{D}} \ddot{\mathbf{D}}^T \right)^{-1} \right) \\ &= \det \left( \ddot{\mathbf{V}}_1^T \mathbf{C}^T \mathbf{C} \ddot{\mathbf{V}}_1 \right) \\ &= \det \left( \mathbf{C} \ddot{\mathbf{V}}_1 \right)^2 \\ &\leq \prod_{k=1}^{K_0} \left( \mathbf{C} \ddot{\mathbf{V}}_1 \right)_{k,k}^2 \\ &\leq \prod_{k=1}^{K_0} \left( \mathbf{C} \mathbf{C}^T \right)_{k,k}^2 \left( \ddot{\mathbf{V}}_1^T \ddot{\mathbf{V}}_1 \right)_{k,k}^2 \\ &= 1, \end{aligned} \quad (\text{A.103})$$

where  $\ddot{\mathbf{V}}_1 = [\ddot{\mathbf{v}}_1, \ddot{\mathbf{v}}_2, \dots, \ddot{\mathbf{v}}_{K_0}] \in \mathbb{R}^{K_1 \times K_0}$ . The last equality holds because of  $\mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}$  and  $\ddot{\mathbf{V}}_1^T \ddot{\mathbf{V}}_1 = \mathbf{I}_{K_0}$ . This establishes inequality (A.101) and the equality holds if and only if  $K_0 = K_1$  or  $\mathbf{C} \ddot{\mathbf{V}}_1 = \mathbf{I}_{K_0}$ .

Similarly, we get inequality (A.99):

$$\ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right) \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}}^2 \mathbf{C}^T \right) \right). \quad (\text{A.104})$$

By Jensen's inequality, we have

$$\left\langle \phi(\hat{y}_k) \right\rangle_{\hat{\mathbf{x}}}^2 \leq \left\langle \phi(\hat{y}_k)^2 \right\rangle_{\hat{\mathbf{x}}}, \forall k = 1, \dots, K_1. \quad (\text{A.105})$$

Then it follows from (A.105) that inequality (A.100) holds:

$$\frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}}^2 \mathbf{C}^T \right) \right) \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right). \quad (\text{A.106})$$

This completes the proof of **Proposition 2**.  $\square$

By **Proposition 2**, if  $K_0 = K_1$  then we get

$$\frac{1}{2} \left\langle \ln \left( \det \left( \hat{\Phi} \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq \frac{1}{2} \ln \left( \det \left( \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \right) \right), \quad (\text{A.107})$$

$$\left\langle \ln \left( \det \left( \Phi \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq \ln \left( \det \left( \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}} \right) \right) \quad (\text{A.108})$$

$$= \frac{1}{2} \ln \left( \det \left( \left\langle \Phi \right\rangle_{\hat{\mathbf{x}}}^2 \right) \right) \quad (\text{A.109})$$

$$\leq \frac{1}{2} \ln \left( \det \left( \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \right) \right), \quad (\text{A.110})$$

$$\ln \left( \det \left( \Phi \right) \right) = \frac{1}{2} \ln \left( \det \left( \hat{\Phi} \right) \right). \quad (\text{A.111})$$

On the other hand, it follows from (A.81) and **Proposition 2** that

$$\left\langle \ln \left( \det \left( \mathbf{C} \Phi \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq -Q \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right), \quad (\text{A.112})$$

$$\left\langle \ln \left( \det \left( \mathbf{C} \Phi \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}} \leq -\hat{Q} \leq \frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}} \mathbf{C}^T \right) \right). \quad (\text{A.113})$$

Hence we can see that  $\hat{Q}$  is close to  $Q$  (see A.81). Moreover, it follows from the Cauchy–Schwarz inequality that

$$\left\langle \left( \Phi \right)_{k,k} \right\rangle_{\hat{\mathbf{x}}} = \left\langle \phi \left( \hat{y}_k \right) \right\rangle_{\hat{y}_k} \leq \left( \int \phi \left( \hat{y}_k \right)^2 d\hat{y}_k \int p \left( \hat{y}_k \right)^2 d\hat{y}_k \right)^{1/2}, \quad (\text{A.114})$$

where  $k = 1, \dots, K_1$ , the equality holds if and only if the following holds:

$$p \left( \hat{y}_k \right) = \frac{\phi \left( \hat{y}_k \right)}{\int \phi \left( \hat{y}_k \right) d\hat{y}_k}, k = 1, \dots, K_1, \quad (\text{A.115})$$

which is the similar to Eq. (A.63).

Since  $I(X; R) = I(Y; R)$  (see **Proposition 1**), by maximizing  $I(X; R)$  we hope the equality in inequality (A.61) and equality (A.63) hold, at least approximatively. On the other hand, let

$$\mathbf{C}^{opt} = \arg \min_{\mathbf{C}} Q[\mathbf{C}] = \arg \max_{\mathbf{C}} \left( \left\langle \ln \left( \det \left( \mathbf{C} \hat{\Phi} \mathbf{C}^T \right) \right) \right\rangle_{\hat{\mathbf{x}}} \right), \quad (\text{A.116})$$

$$\hat{\mathbf{C}}^{opt} = \arg \min_{\mathbf{C}} \hat{Q}[\mathbf{C}] = \arg \max_{\mathbf{C}} \left( \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}}^2 \mathbf{C}^T \right) \right) \right), \quad (\text{A.117})$$

$\mathbf{C}^{opt}$  and  $\hat{\mathbf{C}}^{opt}$  make (A.63) and (A.115) to hold true, which implies that they are the same optimal solution:  $\mathbf{C}^{opt} = \hat{\mathbf{C}}^{opt}$ .

Therefore, we can use the following objective function  $\hat{Q}[\mathbf{C}]$  as a substitute for  $Q[\mathbf{C}]$  and write the optimization problem as:

$$\text{minimize } \hat{Q}[\mathbf{C}] = -\frac{1}{2} \ln \left( \det \left( \mathbf{C} \left\langle \hat{\Phi} \right\rangle_{\hat{\mathbf{x}}}^2 \mathbf{C}^T \right) \right), \quad (\text{A.118})$$

$$\text{subject to } \mathbf{C} \mathbf{C}^T = \mathbf{I}_{K_0}. \quad (\text{A.119})$$

The update rule (A.90) may also apply here and a modified algorithm similar to Algorithm 1 may be used for parameter learning.

## A.5 SUPPLEMENTARY EXPERIMENTS

### A.5.1 QUANTITATIVE METHODS FOR COMPARISON

To quantify the efficiency of learning representations by the above algorithms, we calculate the coefficient entropy (CFE) for estimating coding cost as follows (Lewicki & Olshausen, 1999; Lewicki & Sejnowski, 2000):

$$\tilde{y}_k = \zeta \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}, k = 1, \dots, K_1, \quad (\text{A.120})$$

$$\zeta = \frac{K_1}{\sum_{k=1}^{K_1} \|\tilde{\mathbf{w}}_k\|}, \quad (\text{A.121})$$



where  $\tilde{\mathbf{x}}$  is defined by Eq. (A.68), and  $\tilde{\mathbf{w}}_k$  is the corresponding optimal filter. To estimate the probability density of coefficients  $q_k(\tilde{y}_k)$  ( $k = 1, \dots, K_1$ ) from the  $M$  training samples, we apply the kernel density estimation for  $q_k(\tilde{y}_k)$  and use a normal kernel with an adaptive optimal window width. Then we define the CFE  $h$  as

$$h = \frac{1}{K_1} \sum_{k=1}^{K_1} H_k(\tilde{Y}_k), \quad (\text{A.122})$$

$$H_k(\tilde{Y}_k) = -\Delta \sum_n q_k(n\Delta) \log_2 q_k(n\Delta), \quad (\text{A.123})$$

where  $q_k(\tilde{y}_k)$  is quantized as discrete  $q_k(n\Delta)$  and  $\Delta$  is the step size.

Methods such as IICA and SRBM as well as our methods have feedforward structures in which information is transferred directly through a nonlinear function, e.g., the sigmoid function. We can use the amount of transmitted information to measure the results learned by these methods. Consider a neural population with  $N$  neurons, which is a stochastic system with nonlinear transfer functions. We chose a sigmoidal transfer function and Gaussian noise with standard deviation set to 1 as the system noise. In this case, from (1), (A.8) and (A.11), we see that the approximate MI  $I_G$  is equivalent to the case of the Poisson neuron model. It follows from (A.70)–(A.82) that

$$I(X; R) = I(\tilde{X}; R) = H(\tilde{X}) - H(\tilde{X}|R) \simeq \tilde{I}_G = H(\tilde{X}) - h_1, \quad (\text{A.124})$$

$$H(\tilde{X}|R) \simeq h_1 = -\frac{1}{2} \left\langle \ln \left( \det \left( \frac{1}{2\pi e} \left( NK_0^{-1} \mathbf{C} \hat{\Phi} \mathbf{C}^T + \mathbf{I}_{K_0} \right) \right) \right) \right\rangle_{\tilde{\mathbf{x}}}, \quad (\text{A.125})$$

where we set  $N = 10^6$ . A good representation should make the MI  $I(X; R)$  as big as possible. Equivalently, for the same inputs, a good representation should make the conditional entropy (CDE)  $H(\tilde{X}|R)$  (or  $h_1$ ) as small as possible.

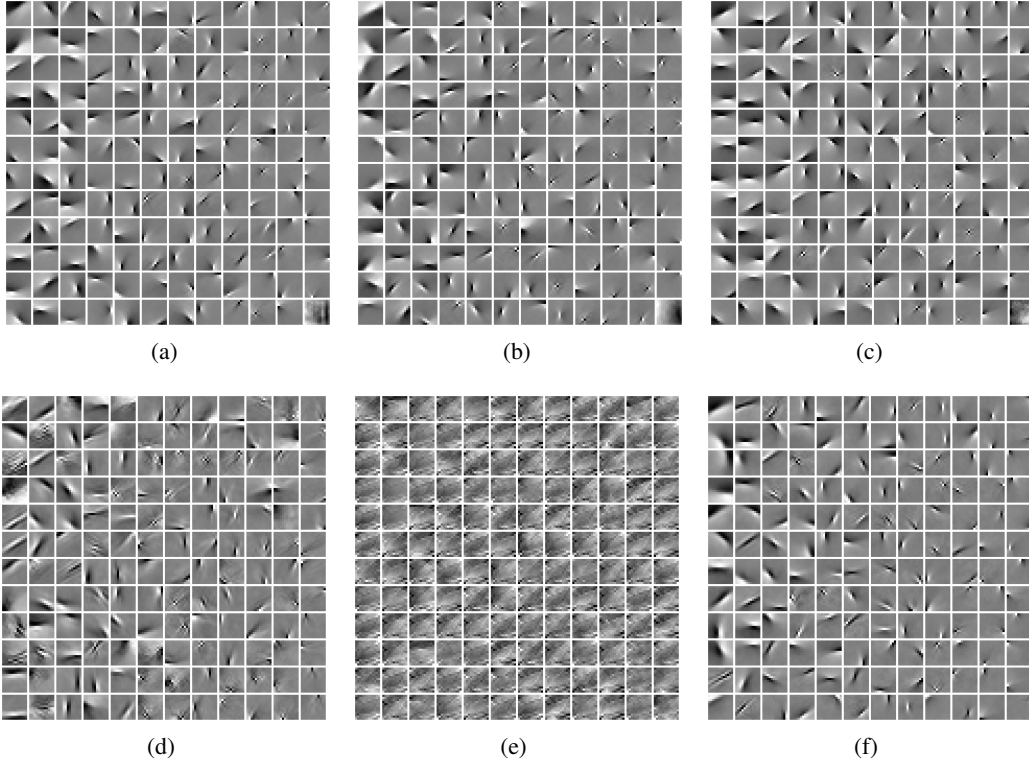


Figure 4: Comparison of basis vectors obtained by our method and other methods. Panel (a)–(e) correspond to panel (a)–(e) in Figure 2, where the basis vectors are given by (A.130). The basis vectors in panel (f) are learned by MBDL and given by (A.127).

### A.5.2 COMPARISON OF BASIS VECTORS

We compared our algorithm with an up-to-date sparse coding algorithm, the mini-batch dictionary learning (MBDL) as given in (Mairal et al., 2009; 2010) and integrated in Python library, i.e. scikit-learn. The input data was the same as the above, i.e.  $10^5$  nature image patches preprocessed by the ZCA whitening filters.

We denote the optimal dictionary learned by MBDL as  $\tilde{\mathbf{B}} \in \mathbb{R}^{K \times K_1}$  for which each column represents a basis vector. Now we have

$$\mathbf{x} \approx \mathbf{U}\Sigma^{1/2}\mathbf{U}^T\tilde{\mathbf{B}}\mathbf{y} = \tilde{\mathbf{B}}\mathbf{y}, \quad (\text{A.126})$$

$$\tilde{\mathbf{B}} = \mathbf{U}\Sigma^{1/2}\mathbf{U}^T\tilde{\mathbf{B}}, \quad (\text{A.127})$$

where  $\mathbf{y} = (y_1, \dots, y_{K_1})^T$  is the coefficient vector.

Similarly, we can obtain a dictionary from the filter matrix  $\mathbf{C}$ . Suppose  $\text{rank}(\mathbf{C}) = K_0 \leq K_1$ , then it follows from (A.64) that

$$\hat{\mathbf{x}} = \left(a\mathbf{C}\mathbf{C}^T\right)^{-1}\mathbf{C}\mathbf{y}. \quad (\text{A.128})$$

By (A.66) and (A.128), we get

$$\mathbf{x} \approx \mathbf{B}\mathbf{y} = a\mathbf{B}\mathbf{C}^T\Sigma_0^{-1/2}\mathbf{U}_0^T\mathbf{x}, \quad (\text{A.129})$$

$$\mathbf{B} = a^{-1}\mathbf{U}_0\Sigma_0^{1/2}\left(\mathbf{C}\mathbf{C}^T\right)^{-1}\mathbf{C} = [\mathbf{b}_1, \dots, \mathbf{b}_{K_1}], \quad (\text{A.130})$$

where  $\mathbf{y} = \mathbf{W}^T\mathbf{x} = a\mathbf{C}^T\Sigma_0^{-1/2}\mathbf{U}_0^T\mathbf{x}$ , the vectors  $\mathbf{b}_1, \dots, \mathbf{b}_{K_1}$  can be regarded as the basis vectors and the strict equality holds when  $K_0 = K_1 = K$ . Recall that  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M] = \mathbf{U}\tilde{\mathbf{V}}^T$  (see Eq. A.49) and  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_M] = \mathbf{W}^T\mathbf{X} = a\sqrt{M-1}\mathbf{C}^T\tilde{\mathbf{V}}_0^T$ , then we get  $\tilde{\mathbf{X}} = \mathbf{B}\mathbf{Y} = \sqrt{M-1}\mathbf{U}_0\Sigma_0^{1/2}\tilde{\mathbf{V}}_0^T \approx \mathbf{X}$ . Hence, Eq. (A.129) holds.

The basis vectors shown in Figure 4(a)–4(e) correspond to filters in Figure 2(a)–2(e). And Figure 4(f) illustrates the optimal dictionary  $\tilde{\mathbf{B}}$  learned by MBDL, where we set the regularization parameter as  $\lambda = 1.2/\sqrt{K}$ , the batch size as 50 and the total number of iterations to perform as 20000, which took about 3 hours for training. From Figure 4 we see that these basis vectors obtained by the above algorithms have local Gabor-like shapes except for those by SRBM. If  $\text{rank}(\tilde{\mathbf{B}}) = K = K_1$ , then the matrix  $\tilde{\mathbf{B}}^{-T}$  can be regarded as a filter matrix like matrix  $\tilde{\mathbf{C}}$  (see Eq. A.69). However, from the column vector of matrix  $\tilde{\mathbf{B}}^{-T}$  we cannot find any local Gabor-like filter that resembles the filters shown in Figure 2. Our algorithm has less computational cost and a much faster convergence rate than the sparse coding algorithm. Moreover, the sparse coding method involves a dynamic generative model that requires relaxation and is therefore unsuitable for fast inference, whereas the feedforward framework of our model is easy for inference because it only requires evaluating the nonlinear tuning functions.

### A.5.3 LEARNING OVERCOMPLETE BASES

We have trained our model on the Olshausen’s nature image patches with a highly overcomplete setup by optimizing the objective (A.118) by Alg.2 and got Gabor-like filters. The results of 400 typical filters chosen from 1024 output filters are displayed in Figure 5(a) and corresponding base (see Eq. A.130) are shown in Figure 5(b). Here the parameters are  $K_1 = 1024$ ,  $t_{\max} = 100$ ,  $v_1 = 0.4$ ,  $\tau = 0.8$ , and  $\epsilon = 0.98$  (see A.52), from which we got  $\text{rank}(\mathbf{B}) = K_0 = 82$ . Compared to the ICA-like results in Figure 2(a)–2(c), the average size of Gabor-like filters in Figure 5(a) is bigger, indicating that the small noise-like local structures in the images have been filtered out.

We have also trained our model on 60,000 images of handwritten digits from MNIST dataset (LeCun et al., 1998) and the resultant 400 typical optimal filters and bases are shown in Figure 5(c) and Figure 5(d), respectively. All parameters were the same as Figure 5(a) and Figure 5(b):  $K_1 = 1024$ ,  $t_{\max} = 100$ ,  $v_1 = 0.4$ ,  $\tau = 0.8$  and  $\epsilon = 0.98$ , from which we got  $\text{rank}(\mathbf{B}) = K_0 = 183$ . From these figures we can see that the salient features of the input images are reflected in these filters and bases. We could also get the similar overcomplete filters and bases by SRBM and MBDL. However, the results depended sensitively on the choice of parameters and the training took a long time.

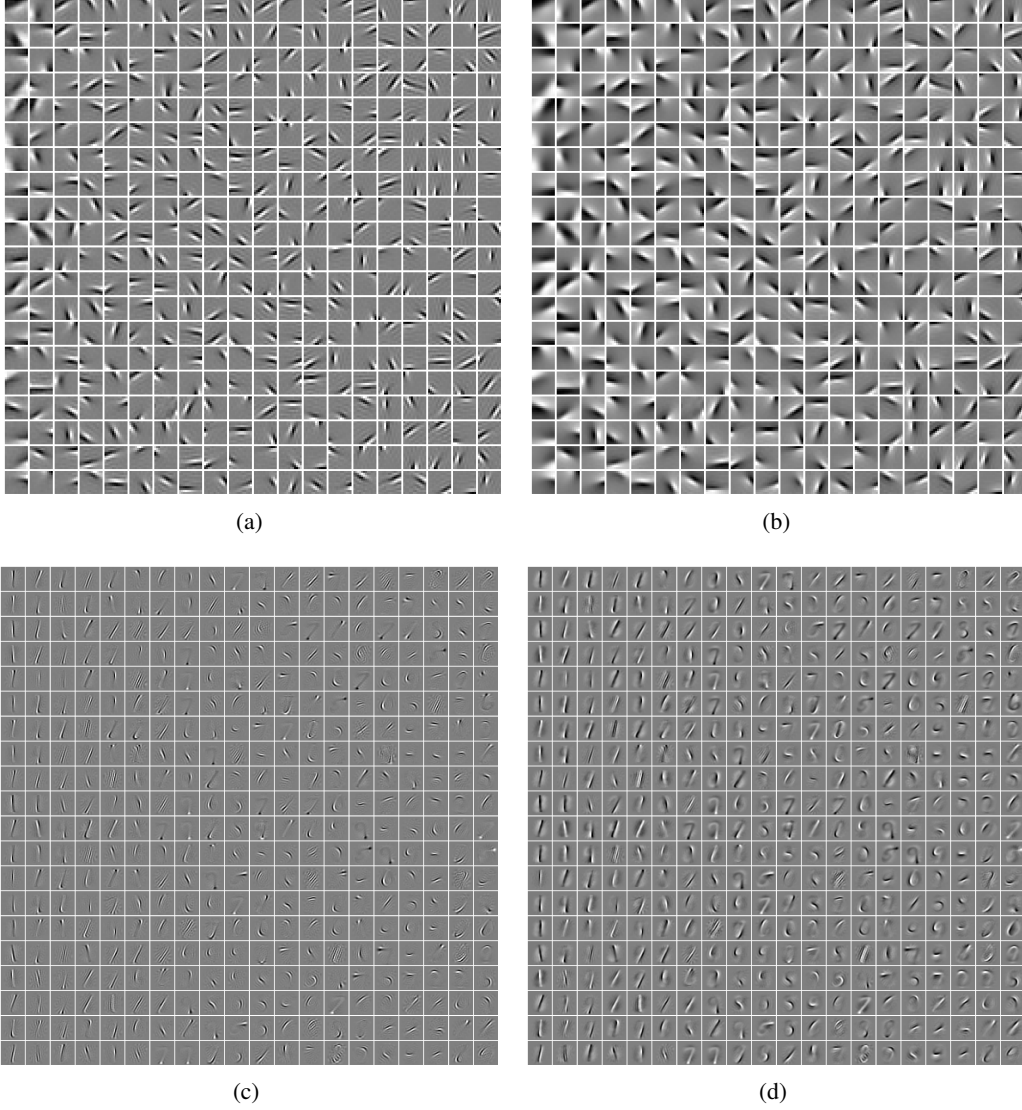


Figure 5: Filters and bases obtained from Olshausen’s image dataset and MNIST dataset by Algorithm 2. **(a)** and **(b)**: 400 typical filters and the corresponding bases obtained from Olshausen’s image dataset, where  $K_0 = 82$  and  $K_1 = 1024$ . **(c)** and **(d)**: 400 typical filters and the corresponding bases obtained from the MNIST dataset, where  $K_0 = 183$  and  $K_1 = 1024$ .

Figure 6 shows that CFE as a function of training time for Alg.2, where Figure 6(a) corresponds to Figure 5(a)-5(b) for learning nature image patches and Figure 6(b) corresponds to Figure 5(c)-5(d) for learning MNIST dataset. We set parameters  $t_{\max} = 100$  and  $\tau = 0.8$  for all experiments and varied parameter  $v_1$  for each experiment, with  $v_1 = 0.2, 0.4, 0.6$  or  $0.8$ . These results indicate a fast convergence rate for training on different datasets. Generally, the convergence is insensitive to the change of parameter  $v_1$ .

We have also performed additional tests on other image datasets and got similar results, confirming the speed and robustness of our learning method. Compared with other methods, e.g., IICA, FICA, MBDL, SRBM or sparse autoencoders etc., our method appeared to be more efficient and robust for unsupervised learning of representations. We also found that complete and overcomplete filters and bases learned by our methods had local Gabor-like shapes while the results by SRBM or MBDL did not have this property.

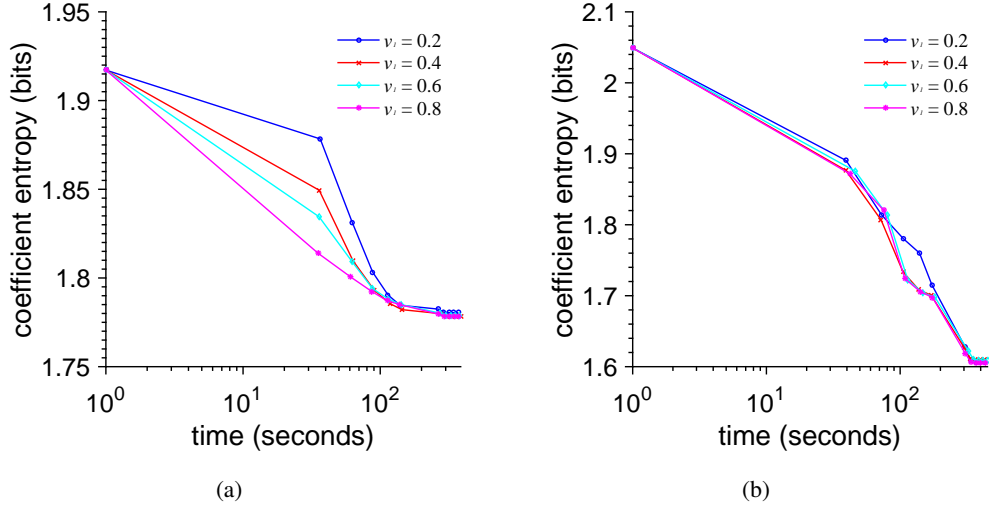


Figure 6: CFE as a function of training time for Alg.2, with  $v_1 = 0.2, 0.4, 0.6$  or  $0.8$ . In all experiments parameters were set to  $t_{\max} = 100$ ,  $t_0 = 50$  and  $\tau = 0.8$ . **(a)**: corresponding to Figure 5(a) or Figure 5(b). **(b)**: corresponding to Figure 5(c) or Figure 5(d).

#### A.5.4 IMAGE DENOISING

Similar to the sparse coding method applied to image denoising (Elad & Aharon, 2006), our method (see Eq. A.130) can also be applied to image denoising, as shown by an example in Figure 7. The filters or bases were learned by using  $7 \times 7$  image patches sampled from the left half of the image, and subsequently used to reconstruct the right half of the image which was distorted by Gaussian noise. A common practice for evaluating the results of image denoising is by looking at the difference between the reconstruction and the original image. If the reconstruction is perfect the difference should look like Gaussian noise. In Figure 7(c) and 7(d) a dictionary (100 bases) was learned by MBDL and orthogonal matching pursuit was used to estimate the sparse solution.<sup>1</sup> For our method (shown in Figure 7(b)), we first get the optimal filters parameter  $\mathbf{W}$ , a low rank matrix ( $K_0 < K$ ), then from the distorted image patches  $\mathbf{x}_m$  ( $m = 1, \dots, M$ ) we get filter outputs  $\mathbf{y}_m = \mathbf{W}^T \mathbf{x}_m$  and the reconstruction  $\tilde{\mathbf{x}}_m = \mathbf{B} \mathbf{y}_m$  (parameters:  $\epsilon = 0.975$  and  $K_0 = K_1 = 14$ ). As can be seen from Figure 7, our method worked better than dictionary learning, although we only used 14 bases compared with 100 bases used by dictionary learning. Our method is also more efficient. We can get better optimal bases  $\mathbf{B}$  by a generative model using our infomax approach (details not shown).

<sup>1</sup>Python source code is available at [http://scikit-learn.org/stable/\\_downloads/plot\\_image\\_denoising.py](http://scikit-learn.org/stable/_downloads/plot_image_denoising.py)

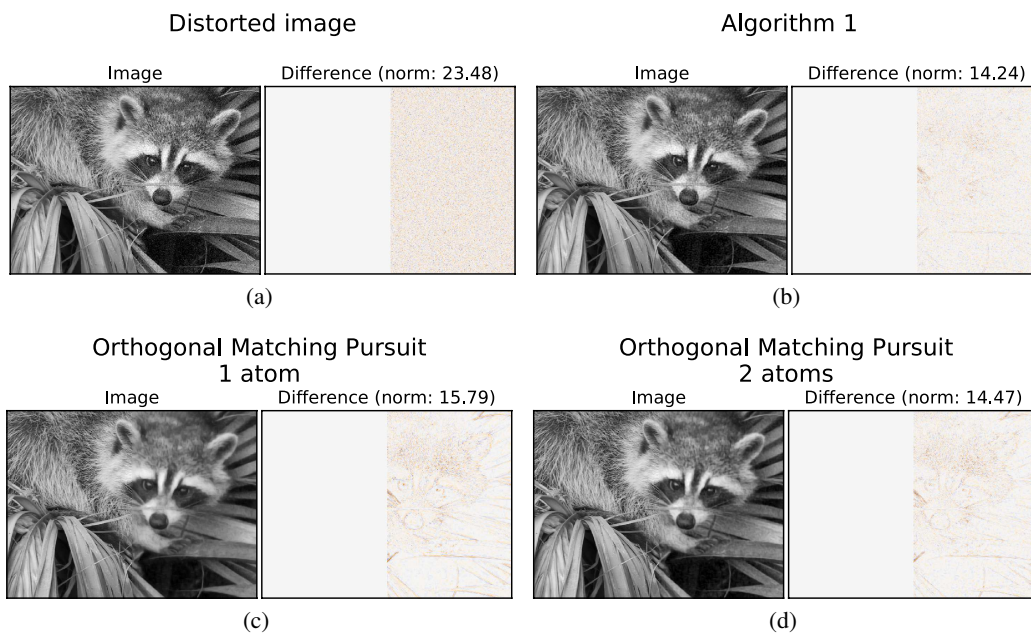


Figure 7: Image denoising. **(a)**: the right half of the original image is distorted by Gaussian noise and the norm of the difference between the distorted image and the original image is 23.48. **(b)**: image denoising by our method (Algorithm 1), with 14 bases used. **(c)** and **(d)**: image denoising using dictionary learning, with 100 bases used.